# SCC/ESCC

## User's Manual

UM010901-0601

ZiLOG Worldwide Headquarters • 910 E. Hamilton Avenue • Campbell, CA 95008
Telephone: 408.558.8500 • Fax: 408.558.8300 • www.ZiLOG.com

This publication is subject to replacement by a later edition. To determine whether a later edition exists, or to request copies of publications, contact:

**ZiLOG Worldwide Headquarters**
910 E. Hamilton Avenue
Campbell, CA 95008
Telephone: 408.558.8500
Fax: 408.558.8300
www.ZiLOG.com

Windows is a registered trademark of Microsoft Corporation.

## Document Disclaimer

©2001 by ZiLOG, Inc. All rights reserved. Information in this publication concerning the devices, applications, or technology described is intended to suggest possible uses and may be superseded. ZiLOG, INC. DOES NOT ASSUME LIABILITY FOR OR PROVIDE A REPRESENTATION OF ACCURACY OF THE INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT. ZiLOG ALSO DOES NOT ASSUME LIABILITY FOR INTELLECTUAL PROPERTY INFRINGEMENT RELATED IN ANY MANNER TO USE OF INFORMATION, DEVICES, OR TECHNOLOGY DESCRIBED HEREIN OR OTHERWISE. Devices sold by ZiLOG, Inc. are covered by warranty and limitation of liability provisions appearing in the ZiLOG, Inc. Terms and Conditions of Sale. ZiLOG, Inc. makes no warranty of merchantability or fitness for any purpose Except with the express written approval of ZiLOG, use of information, devices, or technology as critical components of life support systems is not authorized. No licenses are conveyed, implicitly or otherwise, by this document under any intellectual property rights.

# TABLE OF CONTENTS

**Chapter 1. General Description**

1.1 Introduction ..... 1-1
1.2 SCC's Capabilities ..... 1-2
1.3 Block Diagram ..... 1-4
1.4 Pin Descriptions ..... 1-5
1.4.1 Pins Common to both Z85X30 and Z80X30 ..... 1-7
1.4.2 Pin Descriptions, (Z85X30 Only) ..... 1-8
1.4.3 Pin Descriptions, (Z80X30 Only) ..... 1-9

**Chapter 2. Interfacing the SCC/ESCC**

2.1 Introduction ..... 2-1
2.2 Z80X30 Interface Timing ..... 2-1
2.2.1 Z80X30 Read Cycle Timing ..... 2-2
2.2.2 Z80X30 Write Cycle Timing ..... 2-3
2.2.3 Z80X30 Interrupt Acknowledge Cycle Timing ..... 2-4
2.2.4 Z80X30 Register Access ..... 2-5
2.2.5 Z80C30 Register Enhancement ..... 2-8
2.2.6 Z80230 Register Enhancements ..... 2-8
2.2.7 Z80X30 Reset ..... 2-9
2.3 Z85X30 Interface Timing ..... 2-10
2.3.1 Z85X30 Read Cycle Timing ..... 2-10
2.3.2 Z85X30 Write Cycle Timing ..... 2-11
2.3.3 Z85X30 Interrupt Acknowledge Cycle Timing ..... 2-11
2.3.4 Z85X30 Register Access ..... 2-12
2.3.5 Z85C30 Register Enhancement ..... 2-14
2.3.6 Z85C30/Z85230 Register Enhancements ..... 2-14
2.3.7 Z85X30 Reset ..... 2-15
2.4 Interface Programming ..... 2-15
2.4.1 I/O Programming Introduction ..... 2-15
2.4.2 Polling ..... 2-16
2.4.3 Interrupts ..... 2-16
2.4.4 Interrupt Control ..... 2-17
2.4.5 Daisy-Chain Resolution ..... 2-19
2.4.6 Interrupt Acknowledge ..... 2-21
2.4.7 The Receiver Interrupt ..... 2-21
2.4.8 Transmit Interrupts and Transmit Buffer Empty Bit ..... 2-25
2.4.9 External/Status Interrupts ..... 2-31
2.5 Block/DMA Transfer ..... 2-33
2.5.1 Block Transfers ..... 2-33
2.5.2 DMA Requests ..... 2-36
2.6 Test Functions ..... 2-41
2.6.1 Local Loopback ..... 2-41
2.6.2 Auto Echo ..... 2-41

**Chapter 3. SCC/ESCC Ancillary Support Circuitry**
3.1 Introduction ... 3-1
3.2 Baud Rate Generator ... 3-1
3.3 Data Encoding/Decoding ... 3-4
3.4 DPLL Digital Phase-Locked Loop ... 3-7
3.4.1 DPLL Operation in the NRZI Mode ... 3-8
3.4.2 DPLL Operation in the FM Modes ... 3-9
3.4.3 DPLL Operation in the Manchester Mode ... 3-10
3.4.4 Transmit Clock Counter (ESCC only) ... 3-10
3.5 Clock Selection ... 3-11
3.6 Crystal Oscillator ... 3-14

**Chapter 4. Data Communication Modes**
4.1 Introduction ... 4-1
4.1.1 Transmit Data Path Description ... 4-1
4.1.2 Receive Data Path Description ... 4-2
4.2 Asynchronous Mode ... 4-3
4.2.1 Asynchronous Transmit ... 4-4
4.2.2 Asynchronous Receive ... 4-6
4.2.3 Asynchronous Initialization ... 4-7
4.3 Byte-Oriented Synchronous Mode ... 4-8
4.3.1 Byte-Oriented Synchronous Transmit ... 4-8
4.3.2 Byte-Oriented Synchronous Receive ... 4-10
4.3.3 Transmitter/Receiver Synchronization ... 4-17
4.4 Bit-Oriented Synchronous (SDLC/HDLC) Mode ... 4-18
4.4.1 SDLC Transmit ... 4-19
4.4.2 SDLC Receive ... 4-22
4.4.3 SDLC Frame Status FIFO ... 4-27
4.4.4 SDLC Loop Mode ... 4-30

**Chapter 5. Register Descriptions**
5.1 Introduction ... 5-1
5.2 Write Registers ... 5-2
5.2.1 Write Register 0 (Command Register) ... 5-2
5.2.2 Write Register 1 (Transmit/Receive Interrupt and Data Transfer Mode Definition) ... 5-4
5.2.3 Write Register 2 (Interrupt Vector) ... 5-7
5.2.4 Write Register 3 (Receive Parameters and Control) ... 5-7
5.2.5 Write Register 4 (Transmit/Receive Miscellaneous Parameters and Modes) ... 5-8
5.2.6 Write Register 5 (Transmit Parameters and Controls) ... 5-9
5.2.7 Write Register 6 (Sync Characters or SDLC Address Field) ... 5-10
5.2.8 Write Register 7 (Sync Character or SDLC Flag) ... 5-11
5.2.9 Write Register 7 Prime (ESCC only) ... 5-12
5.2.10 Write Register 7 Prime (85C30 only) ... 5-13
5.2.11 Write Register 8 (Transmit Buffer) ... 5-13
5.2.12 Write Register 9 (Master Interrupt Control) ... 5-14
5.2.13 Write Register 10 (Miscellaneous Transmitter/Receiver Control Bits) ... 5-15
5.2.14 Write Register 11 (Clock Mode Control) ... 5-17
5.2.15 Write Register 12 (Lower Byte of Baud Rate Generator Time Constant) ... 5-18
5.2.16 Write Register 13 (Upper Byte of Baud Rate Generator Time Constant) ... 5-19
5.2.17 Write Register 14 (Miscellaneous Control Bits) ... 5-19
5.2.18 Write Register 15 (External/Status Interrupt Control) ... 5-20
5.3 Read Registers ... 5-21
5.3.1 Read Register 0 (Transmit/Receive Buffer Status and External Status) ... 5-21
5.3.2 Read Register 1 ... 5-23
5.3.3 Read Register 2 ... 5-24
5.3.4 Read Register 3 ... 5-25
5.3.5 Read Register 4 (ESCC and 85C30 Only) ... 5-25
5.3.6 Read Register 5 (ESCC and 85C30 Only) ... 5-25

5.3.7 Read Register 6 (Not on NMOS) ..... 5-25
5.3.8 Read Register 7 (Not on NMOS) ..... 5-25
5.3.9 Read Register 8 ..... 5-26
5.3.10 Read Register 9 (ESCC and 85C30 Only) ..... 5-26
5.3.11 Read Register 10 ..... 5-26
5.3.12 Read Register 11 (ESCC and 85C30 Only) ..... 5-27
5.3.13 Read Register 12 ..... 5-27
5.3.14 Read Register 13 ..... 5-27
5.3.15 Read Register 14 (ESCC and 85C30 Only) ..... 5-27
5.3.16 Read Register 15 ..... 5-27

**Chapter 6. Application Notes**

Interfacing Z80® CPUs to the Z8500 Peripheral Family ..... 6-1
The Z180™ Interfaced with the SCC at MHZ..... 6-34
The Zilog Datacom Family with the 80186 CPU ..... 6-59
SCC in Binary Synchronous Communications..... 6-79
Serial Communication Controller (SCC™): SDLC Mode of Operation ..... 6-93
Using SCC with Z8000 in SDLC Protocol6-105
Boost Your System Performance Using The Zilog ESCC™ ..... 6-117
Technical Considerations When Implementing LocalTalk Link Access Protocol ..... 6-131
On-Chip Oscillator Design..... 6-151

**Chapter 7. Questions and Answers**

Zilog SCC Z8030/Z8530 Questions and Answers..... 7-1
Zilog ESCC™ Controller Questions and Answers ..... 7-11

# LIST OF FIGURES

**Chapter 1**


Figure 1-1. SCC Block Diagram ..... 1-4
Figure 1-2. Z85X30 Pin Functions ..... 1-5
Figure 1-3. Z80X30 Pin Functions ..... 1-6
Figure 1-4. Z85X30 DIP Pin Assignments ..... 1-6
Figure 1-5. Z85X30 PLCC Pin Assignments ..... 1-6
Figure 1-6. Z80X30 DIP Pin Assignments ..... 1-7
Figure 1-7. Z80X30 PLCC Pin Assignments ..... 1-7


**Chapter 2**


Figure 2-1. Z80X30 Read Cycle ..... 2-2
Figure 2-2. Z80X30 Write Cycle ..... 2-3
Figure 2-3. Z80X30 Interrupt Acknowledge Cycle ..... 2-4
Figure 2-4. Write Register 7 Prime (WR7') ..... 2-8
Figure 2-5. Z85X30 Read Cycle Timing ..... 2-10
Figure 2-6. Z85X30 Write Cycle Timing ..... 2-11
Figure 2-7. Z85X30 Interrupt Acknowledge Cycle Timing ..... 2-11
Figure 2-8a. Write Register 7 Prime (WR7') for the 85230 ..... 2-14
Figure 2-8b. Write Register 7 Prime for the 85C30 ..... 2-14
Figure 2-9. ESCC Interrupt Sources ..... 2-16
Figure 2-10. Peripheral Interrupt Structure ..... 2-17
Figure 2-11. Internal Priority Resolution ..... 2-17
Figure 2-12. RR3 Interrupt Pending Bits ..... 2-18
Figure 2-13. Interrupt Flow Chart (for each interrupt source). ..... 2-20
Figure 2-14. Write Register 1 Receive Interrupt Mode Control ..... 2-22
Figure 2-15. Special Conditions Interrupt Service Flow ..... 2-24
Figure 2-16. Transmit Interrupt Status When WR7' D5=1 For ESCC ..... 2-26
Figure 2-17. Transmit Buffer Empty Bit Status For ESCC For Both WR7' and WR7' D5=0 ..... 2-27
Figure 2-18. Transmit Interrupt Status When WR7' D5=0 For ESCC ..... 2-27
Figure 2-19. TxIP Latching on the ESCC ..... 2-27
Figure 2-20. Operation of TBE, Tx Underrun/EOM and TxIP on NMOS/CMOS. ..... 2-28
Figure 2-21. Operation of TBE, Tx Underrun/EOM and TxIP on ESCC ..... 2-29
Figure 2-22. Flowchart example of processing an end of packet ..... 2-30
Figure 2-23. RR0 External/Status Interrupt Operation ..... 2-31
Figure 2-24. Wait On Transmit Timing ..... 2-34
Figure 2-25. Wait On Transmit Timing ..... 2-34
Figure 2-26. Wait On Receive Timing ..... 2-35

Figure 2-27. Wait On Receive Timing ..... 2-35
Figure 2-28. Transmit Request Assertion ..... 2-36
Figure 2-29. Z80X30 Transmit Request Release ..... 2-37
Figure 2-30. Z85X30 Transmit Request Release ..... 2-37
Figure 2-31. /DTR//REQ Deassertion Timing ..... 2-38
Figure 2-32. DMA Receive Request Assertion ..... 2-39
Figure 2-33. Z80X30 Receive Request Release ..... 2-40
Figure 2-34. Z85X30 Receive Request Release ..... 2-40
Figure 2-35. Local Loopback ..... 2-41
Figure 2-36. Auto Echo ..... 2-41

**Chapter 3**

Figure 3-1. Baud Rate Generator ..... 3-1
Figure 3-2. Baud Rate Generator Start Up ..... 3-2
Figure 3-3. Data Encoding Methods ..... 3-4
Figure 3-4. Manchester Encoding Circuit ..... 3-6
Figure 3-5. Digital Phase-Locked Loop ..... 3-7
Figure 3-6. DPLL in NRZI Mode ..... 3-8
Figure 3-7. DPLL Operating Example (NRZI Mode) ..... 3-9
Figure 3-8. DPLL Operation in the FM Mode ..... 3-9
Figure 3-9. DPLL Transmit Clock Counter Output (ESCC only) ..... 3-11
Figure 3-10. Clock Multiplexer ..... 3-12
Figure 3-11. Async Clock Setup Using an External Crystal ..... 3-13
Figure 3-12. Clock Source Selection ..... 3-13
Figure 3-13. Synchronous Transmission, 1x Clock Rate, FM Data Encoding, using DPLL ..... 3-14

**Chapter 4**

Figure 4-1. Transmit Data Path ..... 4-1
Figure 4-2. Receive Data Path ..... 4-2
Figure 4-3. Asynchronous Message Format ..... 4-3
Figure 4-4. Monosync Data Character Format ..... 4-8
Figure 4-5. Sync Character Programming ..... 4-11
Figure 4-6. /SYNC as an Input ..... 4-11
Figure 4-7. /SYNC as an Output ..... 4-12
Figure 4-8. Changing Character Length ..... 4-13
Figure 4-9. Receive CRC Data Path ..... 4-14
Figure 4-10. Transmitter to Receiver Synchronization ..... 4-17
Figure 4-11. SDLC Message Format ..... 4-18
Figure 4-12. /SYNC as an Output ..... 4-23
Figure 4-13. Changing Character Length ..... 4-24
Figure 4-14. Residue Code 101 Interpretation ..... 4-25
Figure 4-15. SDLC Frame Status FIFO (N/A on NMOS) ..... 4-28
Figure 4-16. SDLC Byte Counting Detail ..... 4-29

**Chapter 5**

Figure 5-1. Write Register 0 in the Z85X30 ..... 5-3
Figure 5-2. Write Register 0 in the Z80X30 ..... 5-3
Figure 5-3. Write Register 1 ..... 5-4
Figure 5-4. Write Register 2 ..... 5-7
Figure 5-5. Write Register 3 ..... 5-7

Figure 5-6. Write Register 4 ..... 5-8
Figure 5-7. Write Register 5 ..... 5-9
Figure 5-8. Write Register 6 ..... 5-11
Figure 5-9. Write Register 7 ..... 5-11
Figure 5-10. Write Register 7 Prime ..... 5-12
Figure 5-10a. Write Register 7 Prime (WR7') ..... 5-13
Figure 5-11. Write Register 9 ..... 5-14
Figure 5-12. Write Register 10 ..... 5-15
Figure 5-13. NRZ (NRZI), FM1 (FM0) Timing ..... 5-16
Figure 5-14. Write Register 11 ..... 5-17
Figure 5-15. Write Register 12 ..... 5-18
Figure 5-16. Write Register 13 ..... 5-19
Figure 5-17. Write Register 14 ..... 5-19
Figure 5-18. Write Register 15 ..... 5-20
Figure 5-19. Read Register 0 ..... 5-21
Figure 5-20. Read Register 1 ..... 5-23
Figure 5-21. Read Register 2 ..... 5-25
Figure 5-22. Read Register 3 ..... 5-25
Figure 5-23. Read Register 6 (Not on NMOS) ..... 5-25
Figure 5-24. Read Register 7 (Not on NMOS) ..... 5-26
Figure 5-25. Read Register 10 ..... 5-26
Figure 5-26. Read Register 12 ..... 5-27
Figure 5-27. Read Register 13 ..... 5-27
Figure 5-28. Read Register 15 ..... 5-27

# LIST OF TABLES

**Chapter 2**

Table 2-1. Z80X30 Register Map (Shift Left Mode) ..... 2-6
Table 2-2. Z80X30 Register Map (Shift Right Mode) ..... 2-7
Table 2-3. Z80230 SDLC/HDLC Enhancement Options ..... 2-8
Table 2-4. Z80X30 Register Reset Values ..... 2-9
Table 2-5. Z85X30 Register Map ..... 2-13
Table 2-6. Z85C30/Z85230 Register Enhancement Options ..... 2-14
Table 2-7. Z85X30 Register Reset Value ..... 2-15
Table 2-8. Interrupt Source Priority ..... 2-16
Table 2-9. Interrupt Vector Modification ..... 2-19

**Chapter 3**

Table 3-1. Baud Rates for 2.4576 MHz Clock and 16x Clock Factor ..... 3-3

**Chapter 4**

Table 4-1. Write Register Bits Ignored in Asynchronous Mode ..... 4-4
Table 4-2. Transmit Bits per Character ..... 4-5
Table 4-3. Initialization Sequence Asynchronous Mode ..... 4-7
Table 4-4. Registers Used in Character-Oriented Modes ..... 4-9
Table 4-5. Transmitter Initialization in Character- Oriented Mode ..... 4-10
Table 4-6. Sync Character Length Selection ..... 4-11
Table 4-7. Enabling and Disabling CRC ..... 4-16
Table 4-8. Initializing the Receiver in Character-Oriented Mode ..... 4-17
Table 4-9. ESCC Action Taken on Tx Underrun ..... 4-20
Table 4-10. Residue Codes ..... 4-24
Table 4-11. Initializing in SDLC Mode ..... 4-26
Table 4-12. SDLC Loop Mode Initialization ..... 4-32

**Chapter 5**

Table 5-1. SCC Write Registers ..... 5-1
Table 5-2. SCC Read Registers ..... 5-1
Table 5-3. Z85X30 Register Map ..... 5-5
Table 5-4. Receive Bits per Character ..... 5-7
Table 5-5. Transmit Bits per Character ..... 5-10
Table 5-6. Interrupt Vector Modification ..... 5-14
Table 5-7. Data Encoding ..... 5-15

Table 5-8. Receive Clock Source ..... 5-18
Table 5-9. Transmit Clock Source ..... 5-18
Table 5-10. Transmit External Control Selection ..... 5-18
Table 5-11. I-Field Bit Selection (8 Bits Only) ..... 5-24
Table 5-12. Bits per Character Residue Decoding ..... 5-24
Table 5-13. Read Register 7 FIFO Status Decoding ..... 5-26

# CHAPTER 1
GENERAL DESCRIPTION

## 1.1 INTRODUCTION

The Zilog SCC Serial Communication Controller is a dual channel, multiprotocol data communication peripheral designed for use with 8- and 16-bit microprocessors. The SCC functions as a serial-to-parallel, parallel-to-serial converter/controller. The SCC can be software-configured to satisfy a wide variety of serial communications applications. The device contains a variety of new, sophisticated internal functions including on-chip baud rate generators, digital phase-lock loops, and crystal oscillators, which dramatically reduce the need for external logic.

The SCC handles asynchronous formats, synchronous byte-oriented protocols such as IBM® Bisync, and synchronous bit-oriented protocols such as HDLC and IBM SDLC. This versatile device supports virtually any serial data transfer application (telecommunication, LAN, etc.)

The device can generate and check CRC codes in any synchronous mode and can be programmed to check data integrity in various modes. The SCC also has facilities for modem control in both channels. In applications where these controls are not needed, the modem controls can be used for general-purpose I/O.

With access to 14 Write registers and 7 Read registers per channel (the number of the registers varies depending on the version), the user can configure the SCC to handle all synchronous formats regardless of data size, number of stop bits, or parity requirements.

Within each operating mode, the SCC also allows for protocol variations by checking odd or even parity bits, character insertion or deletion, CRC generation, checking break and abort generation and detection, and many other protocol-dependent features.

The SCC/ESCC family consists of the following seven devices;

| | Z-Bus® | Universal-Bus |
|---|---|---|
| NMOS | Z8030 | Z8530 |
| CMOS | Z80C30 | Z85C30 |
| ESCC | Z80230 | Z85230 |
| EMSCC | | Z85233 |

As a convention, use the following words to distinguish the devices throughout this document.

**SCC:** Description applies to all versions.

**NMOS:** Description applies to NMOS version (Z8030/Z8530)

**CMOS:** Description applies to CMOS version (Z80C30/Z85C30)

**ESCC:** Description applies to ESCC (Z80230/Z85230)

**EMSCC:** Description applies to EMSCC (Z85233)

**Z80X30:** Description applies to Z-Bus version of the device (Z8030/Z80C30/Z80230)

**Z85X3X:** Description applies to Universal version of the device (Z8530/Z85C30/Z85230/Z85233)

The Z-Bus version has a multiplexed bus interface and is directly compatible with the Z8000, Z16C00 and 80x86 CPUs. The Universal version has a non-multiplexed bus interface and easily interfaces with virtually any CPU, including the 8080, Z80, 68X00.

## 1.2 SCC'S CAPABILITIES

The NMOS version of the SCC is Zilog's original device. The design is based on the Z80 SIO architecture. If you are familiar with the Z80 SIO, the SCC can be treated as an SIO with support circuitry such as DPLL, BRG, etc. Its features include:

- Two independent full-duplex channels
- Synchronous/Isosynchronous data rates:
  - Up to 1/4 of the PCLK using external clock source.
    Up to 5 Mbits/sec at 20 MHz PCLK (ESCC)
    Up to 4 Mbits/sec at 16 MHz PCLK (CMOS)
    Up to 2 MBits/sec at 8 MHz PCLK (NMOS)
  - Up to 1/8 of the PCLK (up to 1/16 on NMOS) using FM encoding with DPLL
  - Up to 1/16 of the PCLK (up to 1/32 on NMOS) using NRZI encoding with DPLL
- Asynchronous Capabilities
  - 5, 6, 7 or 8 bits/character (capable of handling 4 bits/character or less.)
  - 1, 1.5, or 2 stop bits
  - Odd or even parity
  - Times 1, 16, 32 or 64 clock modes
  - Break generation and detection
  - Parity, overrun and framing error detection
- Byte oriented synchronous capabilities:
  - Internal or external character synchronization
  - One or two sync characters (6 or 8 bits/sync character) in separate registers
  - Automatic Cyclic Redundancy Check (CRC) generation/detection
- SDLC/HDLC capabilities:
  - Abort sequence generation and checking
  - Automatic zero insertion and detection
  - Automatic flag insertion between messages
  - Address field recognition
  - I-field residue handling
  - CRC generation/detection
  - SDLC loop mode with EOP recognition/loop entry and exit
- Receiver FIFO
  ESCC: 8 bytes deep
  NMOS/CMOS: 3 bytes deep
- Transmitter FIFO
  ESCC: 4 bytes deep
  NMOS/CMOS: 1 byte deep
- NRZ, NRZI or FM encoding/decoding. Manchester code decoding (encoding with external logic).
- Baud Rate Generator in each channel
- Digital Phase Locked Loop (DPLL) for clock recovery
- Crystal oscillator

The CMOS version of the SCC is 100% plug in compatible to the NMOS versions of the device, while providing the following additional features:

- Status FIFO
- Software interrupt acknowledge feature
- Enhanced timing specifications
- Faster system clock speed
- Designed in Zilog's Superintegration™ core format
- When the DPLL clock source is external, it can be up to 2x the PCLK, where NMOS allows up to PCLK (32.3 MHz max with 16/20 MHz version).

The Z85C30 CMOS SCC has added new features, while maintaining 100% hardware/software compatibility. It has the following new features:

- New programmable WR7' (write register 7 prime) to enable new features.
- Improvements to support SDLC mode of synchronous communication:
  - Improved functionality to ease sending back-to back frames
  - Automatic SDLC opening Flag transmission*
  - Automatic Tx Underrun/EOM Latch reset in SDLC mode*
  - Automatic /RTS deactivation*
  - TxD pin forced "H" in SDLC NRZI mode after closing flag*
  - Complete CRC reception*
  - Improved response to Abort sequence in status FIFO
  - Automatic Tx CRC generator preset/reset
  - Extended read for write registers*
  - Write data setup timing improvement
- Improved AC timing:
  - Three to 3.5 PCLK access recovery time.
  - Programmable /DTR//REQ timing*
  - Elimination of write data to falling edge of /WR setup time requirement
  - Reduced /INT timing
- Other features include:
  - Extended read function to read back the written value to the write registers*
  - Latching RR0 during read
  - RR0, bit D7 and RR10, bit D6 now has reset defaultvalue.

Some of the features listed above are available by default, and some of them (features with "*") are disabled on default.

ESCC (Enhanced SCC) is pin and software compatible to the CMOS version, with the following additional enhancements.

- Deeper transmit FIFO (4 bytes)
- Deeper receive FIFO (8 bytes)
- Programmable FIFO interrupt and DMA request level
- Seven enhancements to improve SDLC link layer supports:
  - Automatic transmission of the opening flag
  - Automatic reset of Tx Underrun/EOM latch
  - Deactivation of /RTS pin after closing flag
  - Automatic CRC generator preset
  - Complete CRC reception
  - TxD pin automatically forced high with NRZI encoding when using mark idle
  - Status FIFO handles better frames with an ABORT
  - Receive FIFO automatically unlocked for special receive interrupts when using the SDLC status FIFO
- Delayed bus latching for easier microprocessor interface
- New programmable features added with Write Register 7' (WR seven prime)
- Write registers 3, 4, 5 and 10 are now readable
- Read register 0 latched during access
- DPLL counter output available as jitter-free transmitter clock source
- Enhanced /DTR, /RTS deactivation timing

## 1.3 BLOCK DIAGRAM

Figure 1-1 has the block diagram of the SCC. Note that the depth of the FIFO differs depending on the version. The 10X19 SDLC Frame Status FIFO is not available on the NMOS version of the SCC.

Detailed internal signal path will be discussed in Chapter 4.

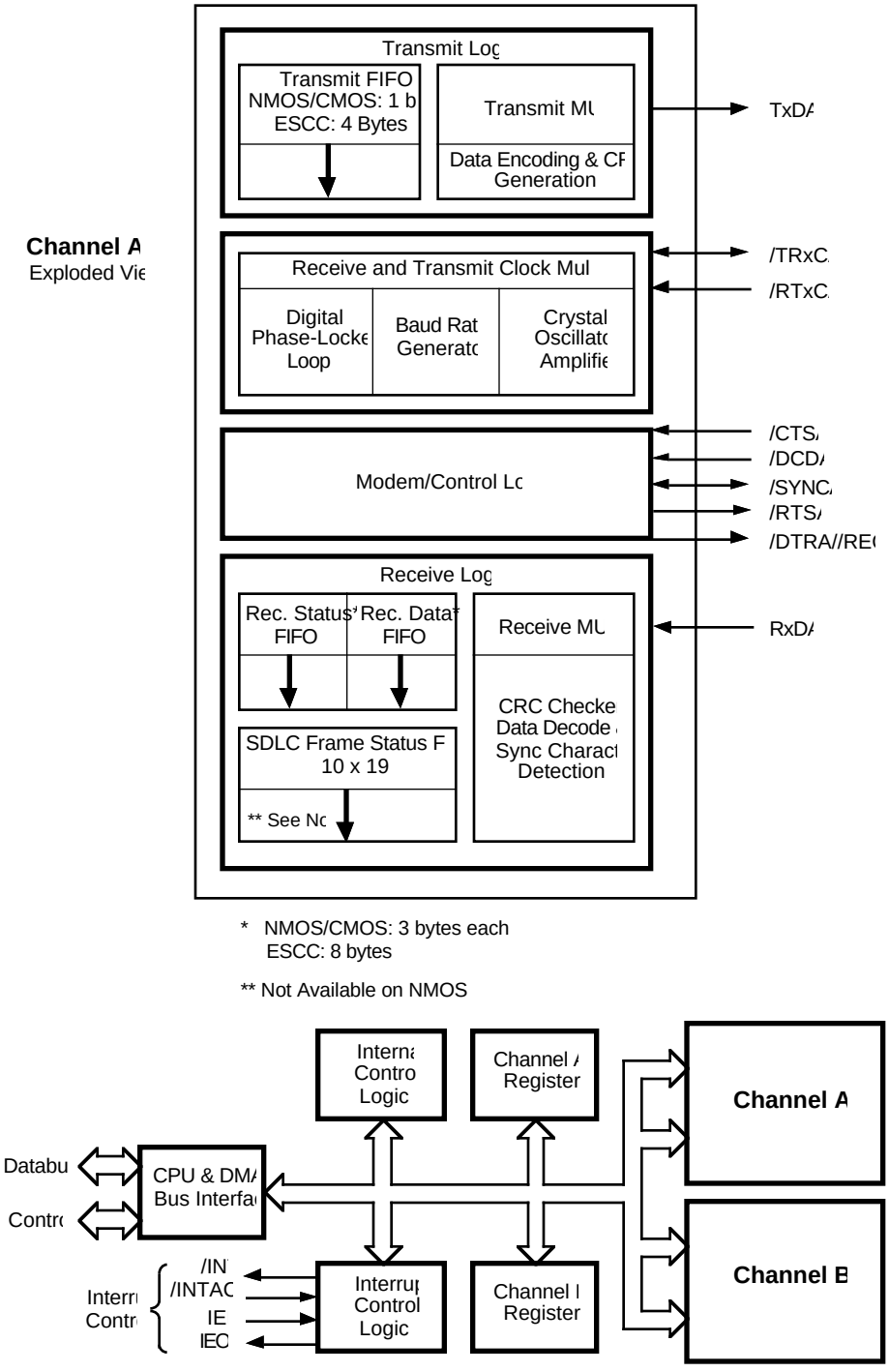



**Figure 1-1. SCC Block Diagram**

## 1.4 PIN DESCRIPTIONS

The SCC pins are divided into seven functional groups: Address/Data, Bus Timing and Reset, Device Control, Interrupt, Serial Data (both channels), Peripheral Control (both channels), and Clocks (both channels). Figures 1-2 and 1-3 show the pins in each functional group for both Z80X30 and Z85X30. Notice the pin functions unique to each bus interface version in the Address/Data group, Bus Timing and Reset group, and Control groups.

The Address/Data group consists of the bidirectional lines used to transfer data between the CPU and the SCC (Addresses in the Z80X30 are latched by /AS). The direction of these lines depends on whether the operation is a Read or Write.

The timing and control groups designate the type of transaction to occur and when it will occur. The interrupt group provides inputs and outputs to conform to the Z-Bus® specifications for handling and prioritizing interrupts. The remaining groups are divided into channel A and channel B groups for serial data (transmit or receive), peripheral control (such as DMA or modem), and the input and output lines for the receive and transmit clocks.

The signal functionality and pin assignments (Figures 1-4 to 1-7) stay constant within the same bus interface group (i.e., Z80X30, Z85X30), except for some timing and/or DC specification differences. For details, please reference the individual product specifications.

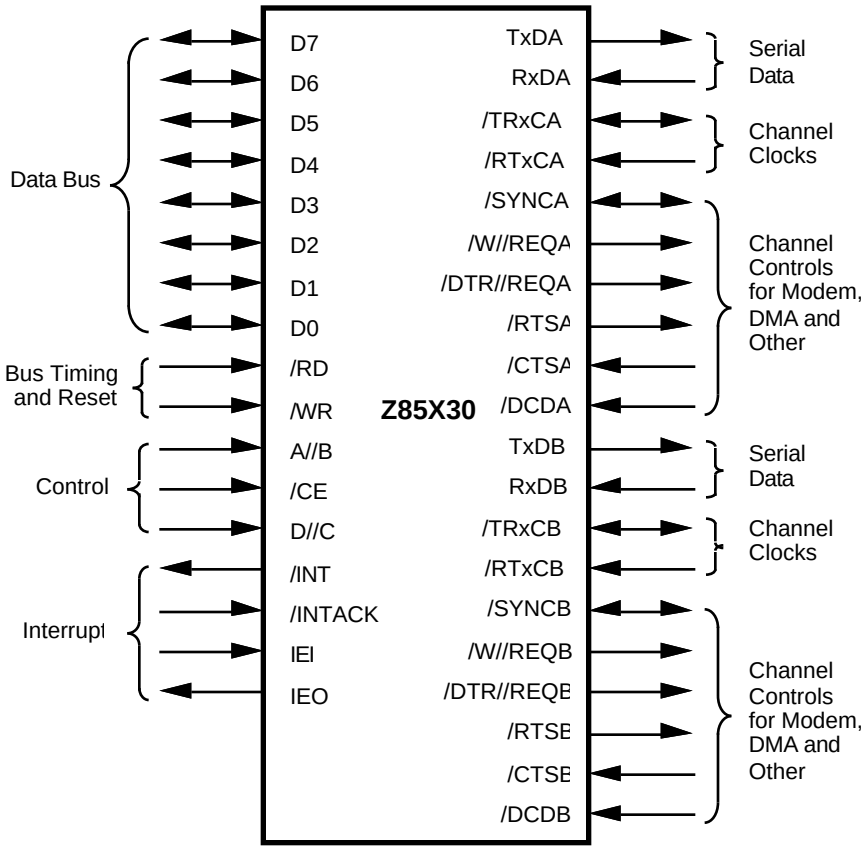



**Figure 1-2. Z85X30 Pin Functions**

## 1.4 PIN DESCRIPTIONS (Continued)

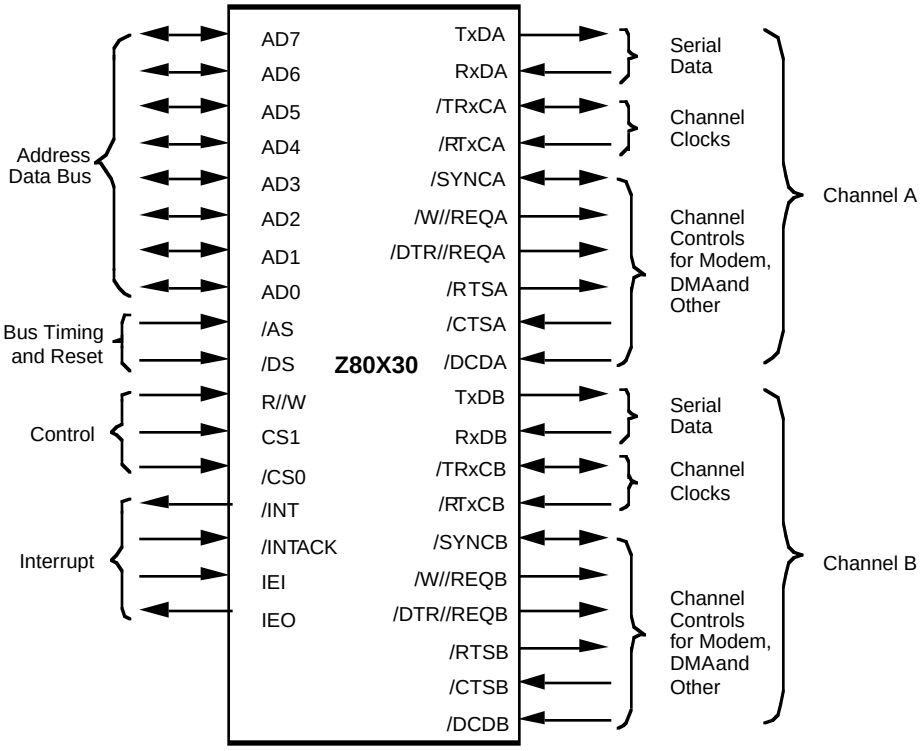



**Figure 1-3. Z80X30 Pin Functions**

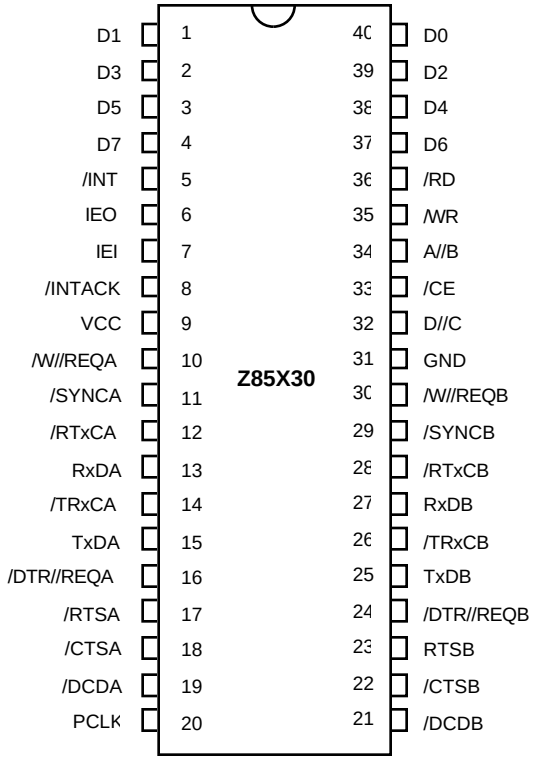



**Figure 1-4. Z85X30 DIP Pin Assignments**

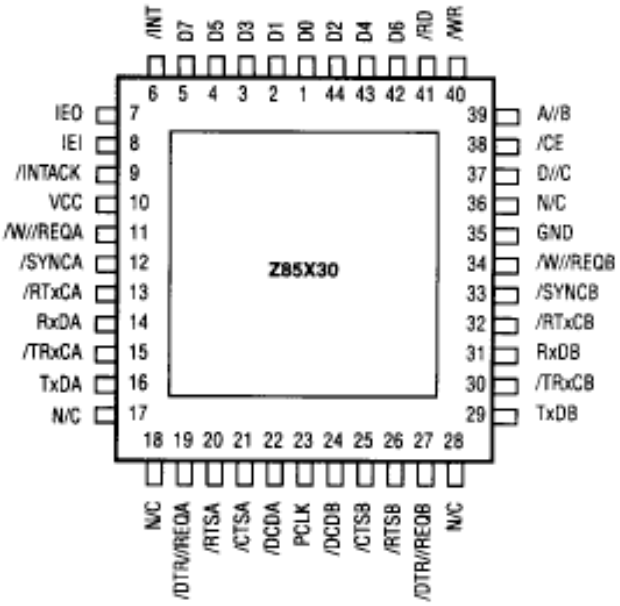



**Figure 1-5. Z85X30 PLCC Pin Assignments**

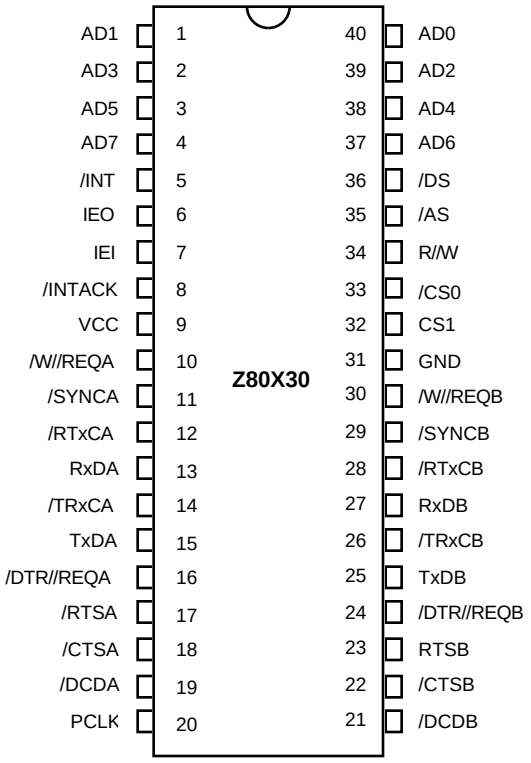



**Figure 1-6. Z80X30 DIP Pin Assignments**

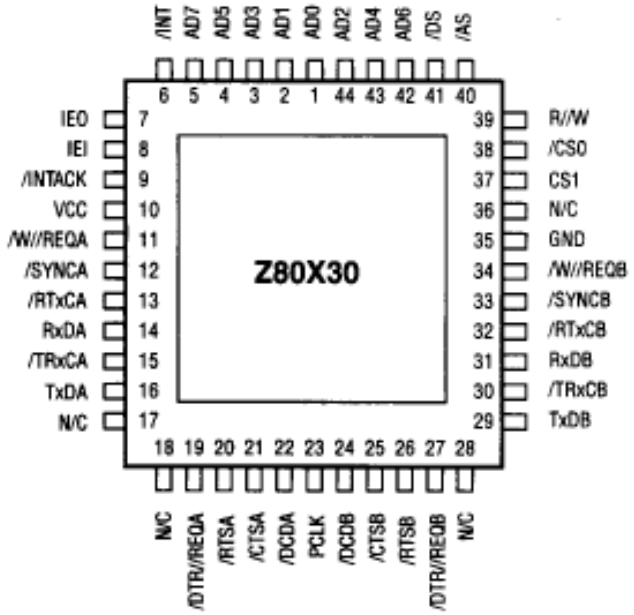



**Figure 1-7. Z80X30 PLCC Pin Assignments**

## 1.4.1 Pins Common to both Z85X30 and Z80X30

**/CTSA, /CTSB.** Clear To Send (inputs, active Low). These pins function as transmitter enables if they are programmed for Auto Enable (WR3, D5=1). A Low on the inputs enables the respective transmitters. If not programmed as Auto Enable, they may be used as general-purpose inputs. Both inputs are Schmitt-trigger buffered to accommodate slow rise-time inputs. The SCC detects pulses on these inputs and can interrupt the CPU on both logic level transitions.

**/DCDA, /DCDB.** Data Carrier Detect (inputs, active Low). These pins function as receiver enables if they are programmed for Auto Enable (WR3, D5=1); otherwise, they are used as general-purpose input pins. Both pins are Schmitt-trigger buffered to accommodate slow rise time signals. The SCC detects pulses on these pins and can interrupt the CPU on both logic level transitions.

**/RTSA, /RTSB.** Request To Send (outputs, active Low). The /RTS pins can be used as general-purpose outputs or with the Auto Enable feature. When used with Auto Enable ON (WR3, D5=1) in asynchronous mode, the /RTS pin goes High after the transmitter is empty. When Auto Enable is OFF, the /RTS pins are used as general-purpose outputs, and, they strictly follow the inverse state of WR5, bit D1.

***ESCC and 85C30:***
***In SDLC mode, the /RTS pins can be programmed to be deasserted when the closing flag of the message clears the TxD pin, if WR7' D2 is set.***

**/SYNCA, /SYNCB**. Synchronization (inputs or outputs, active Low). These pins can act either as inputs, outputs, or part of the crystal oscillator circuit. In the Asynchronous Receive mode (crystal oscillator option not selected), these pins are inputs similar to CTS and DCD. In this mode, transitions on these lines affect the state of the Synchronous/Hunt status bits in Read Register 0 but have no other function.

In External Synchronization mode, with the crystal oscillator not selected, these lines also act as inputs. In this mode, /SYNC is driven Low to receive clock cycles after the last bit in the synchronous character is received. Character assembly begins on the rising edge of the receive clock immediately preceding the activation of SYNC.

In the Internal Synchronization mode (Monosync and Bisync) with the crystal oscillator not selected, these pins act as outputs and are active only during the part of the

## 1.4 PIN DESCRIPTIONS (Continued)

receive clock cycle in which the synchronous condition is not latched. These outputs are active each time a synchronization pattern is recognized (regardless of character boundaries). In SDLC mode, the pins act as outputs and are valid on receipt of a flag. The /SYNC pins switch from input to output when monosync, bisync, or SDLC is programmed in WR4 and sync modes are enabled.

**/DTR//REQA, /DTR//REQB.** Data Terminal Ready/Request (outputs, active Low). These pins are programmable (WR14, D2) to serve either as general-purpose outputs or as DMA Request lines. When programmed for DTR function (WR14 D2=0), these outputs follow the state programmed into the DTR bit of Write Register 5 (WR5 D7). When programmed for Ready mode, these pins serve as DMA Requests for the transmitter.

***ESCC and 85C30:***
***When used as DMA request lines (WR14, D2=1), the timing for the deactivation request can be programmed in the added register, Write Register 7' (WR7') bit D4. If this bit is set, the /DTR//REQ pin is deactivated with the same timing as the /W/REQ pin. If WR7' D4 is reset, the deactivation timing of /DTR//REQ pin is four clock cycles, the same as in the Z85C30.***

**/W//REQA, /W//REQB.** Wait/Request (outputs, open-drain when programmed for Wait function, driven High or Low when programmed for Ready function). These dual-purpose outputs may be programmed as Request lines for a DMA controller or as Wait lines to synchronize the CPU to the SCC data rate. The reset state is Wait.

**RxDA, RxDB.** Receive Data (inputs, active High). These input signals receive serial data at standard TTL levels.

**/RTxCA, /RTxCB.** Receive/Transmit Clocks (inputs, active Low). These pins can be programmed to several modes of operation. In each channel, /RTxC may supply the receive clock, the transmit clock, the clock for the baud rate generator, or the clock for the Digital Phase-Locked Loop. These pins can also be programmed for use with the respective SYNC pins as a crystal oscillator. The receive clock may be 1, 16, 32, or 64 times the data rate in asynchronous modes.

**TxDA, TxDB.** Transmit Data (outputs, active High). These output signals transmit serial data at standard TTL levels.

**/TRxCA, /TRxCB.** Transmit/Receive Clocks (inputs or outputs, active Low). These pins can be programmed in several different modes of operation. /TRxC may supply the receive clock or the transmit clock in the input mode or supply the output of the Transmit Clock Counter (which parallels the Digital Phase-Locked Loop), the crystal oscillator, the baud rate generator, or the transmit clock in the output mode.

**PCLK.** Clock (input). This is the master SCC clock used to synchronize internal signals. PCLK is a TTL level signal. PCLK is not required to have any phase relationship with the master system clock.

**IEI.** Interrupt Enable In (input, active High). IEI is used with IEO to form an interrupt daisy chain when there is more than one interrupt driven device. A high IEI indicates that no other higher priority device has an interrupt under service or is requesting an interrupt.

**IEO.** Interrupt Enable Out (output, active High). IEO is High only if IEI is High and the CPU is not servicing the SCC interrupt or the SCC is not requesting an interrupt (Interrupt Acknowledge cycle only). IEO is connected to the next lower priority device's IEI input and thus inhibits interrupts from lower priority devices.

**/INT.** Interrupt (output, open drain, active Low). This signal is activated when the SCC requests an interrupt. Note that /INT is an open-drain output.

**/INTACK.** Interrupt Acknowledge (input, active Low). This is a strobe which indicates that an interrupt acknowledge cycle is in progress. During this cycle, the SCC interrupt daisy chain is resolved. The device is capable of returning an interrupt vector that may be encoded with the type of interrupt pending. During the acknowledge cycle, if IEI is high, the SCC places the interrupt vector on the databus when /RD goes active. /INTACK is latched by the rising edge of PCLK.

### 1.4.2 Pin Descriptions, (Z85X30 Only)

**D7-D0**. Data bus (bidirectional, tri-state). These lines carry data and commands to and from the Z85X30.

**/CE.** Chip Enable (input, active Low). This signal selects the Z85X30 for a read or write operation.

**/RD.** Read (input, active Low). This signal indicates a read operation and when the Z85X30 is selected, enables the Z85X30's bus drivers. During the Interrupt Acknowledge cycle, /RD gates the interrupt vector onto the bus if the Z85X30 is the highest priority device requesting an interrupt.

**/WR.** Write (input, active Low). When the Z85X30 is selected, this signal indicates a write operation. This indicates that the CPU wants to write command bytes or data to the Z85X30 write registers.

**A//B.** Channel A/Channel B (input). This signal selects the channel in which the read or write operation occurs. High selects channel A and Low selects channel B.

**D//C.** Data/Control Select (input). This signal defines the type of information transferred to or from the Z85X30. High means data is being transferred and Low indicates a command.

### 1.4.3 Pin Descriptions, (Z80X30 Only)

**AD7-AD0**. Address/Data Bus (bidirectional, active High, tri-state). These multiplexed lines carry register addresses to the Z80X30 as well as data or control information to and from the Z80X30.

**R//W.** Read//Write (input, read active High). This signal specifies whether the operation to be performed is a read or a write.

**/CS0.** Chip Select 0 (input, active Low). This signal is latched concurrently with the addresses on AD7-AD0 and must be active for the intended bus transaction to occur.

**CS1.** Chip Select 1 (input, active High). This second select signal must also be active before the intended bus transaction can occur. CS1 must remain active throughout the transaction.

**/DS.** Data Strobe (input, active Low). This signal provides timing for the transfer of data into and out of the Z80X30. If /AS and /DS are both Low, this is interpreted as a reset.

**/AS**. Address Strobe (input, active Low). Address on AD7-AD0 are latched by the rising edge of this signal.

© 1998 by Zilog, Inc. All rights reserved. No part of this document may be copied or reproduced in any form or by any means without the prior written consent of Zilog, Inc. The information in this document is subject to change without notice. Devices sold by Zilog, Inc. are covered by warranty and patent indemnification provisions appearing in Zilog, Inc. Terms and Conditions of Sale only.

ZILOG, INC. MAKES NO WARRANTY, EXPRESS, STATUTORY, IMPLIED OR BY DESCRIPTION, REGARDING THE INFORMATION SET FORTH HEREIN OR REGARDING THE FREEDOM OF THE DESCRIBED DEVICES FROM INTELLECTUAL PROPERTY INFRINGEMENT. ZILOG, INC. MAKES NO WARRANTY OF MERCHANTABILITY OR FITNESS FOR ANY PURPOSE.

Zilog, Inc. shall not be responsible for any errors that may appear in this document. Zilog, Inc. makes no commitment to update or keep current the information contained in this document.

Zilog's products are not authorized for use as critical components in life support devices or systems unless a specific written agreement pertaining to such intended use is executed between the customer and Zilog prior to use. Life support devices or systems are those which are intended for surgical implantation into the body, or which sustains life whose failure to perform, when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in significant injury to the user.

Zilog, Inc. 210 East Hacienda Ave.
Campbell, CA 95008-6600
Telephone (408) 370-8000
FAX 408 370-8056
Internet: http://www.zilog.com

# CHAPTER 2
## INTERFACING THE SCC/ESCC

### 2.1 INTRODUCTION

This chapter covers the system interface requirements with the SCC. Timing requirements for both devices are described in a general sense here, and the user should refer to the SCC Product Specification for detailed AC/DC parametric requirements.

The ESCC and the 85C30 have an additional register, Write Register Seven Prime (WR7'). Its features include the ability to read WR3, WR4, WR5, WR7', and WR10. Both the ESCC and the 85C30 have the ability to deassert the /DTR//REG pin quickly to ease DMA interface design. Additionally, the Z85230 features a relaxed requirement for a valid data bus when the /WR pin goes Low. The effects of the deeper data FIFOs should be considered when writing the interrupt service routines. The user should read the sections which follow for details on these features.

### 2.2 Z80X30 INTERFACE TIMING

The Z-Bus® compatible SCC is suited for system applications with multiplexed address/data buses similar to the Z8®, Z8000®, and Z280®.

Two control signals, /AS and /DS, are used by the Z80X30 to time bus transactions. In addition, four other control signals (/CS0, CS1, R//W, and /INTACK) are used to control the type of bus transaction that occurs. A bus transaction is initiated by /AS; the rising edge latches the register address on the Address/Data bus and the state of /INTACK and /CS0.

In addition to timing bus transactions, /AS is used by the interrupt section to set the Interrupt Pending (IP) bits. Because of this, /AS must be kept cycling for the interrupt section to function properly.

The Z80X30 generates internal control signals in response to a register access. Since /AS and /DS have no phase relationship with PCLK, the circuit generating these internal control signals provides time for metastable conditions to disappear. This results in a recovery time related to PCLK.

This recovery time applies only to transactions involving the Z80X30, and any intervening transactions are ignored. This recovery time is four PCLK cycles, measured from the falling edge of /DS of one access to the SCC, to the falling edge of /DS for a subsequent access.

## 2.2 Z80X30 INTERFACE TIMING (Continued)

### 2.2.1 Z80X30 Read Cycle Timing

The read cycle timing for the Z80X30 is shown in Figure 2-1. The register address on AD7-AD0, as well as the state of /CS0 and /INTACK, are latched by the rising edge of /AS. R//W must be High before /DS falls to indicate a read cycle. The Z80X30 data bus drivers are enabled while CS1 is High and /DS is Low.

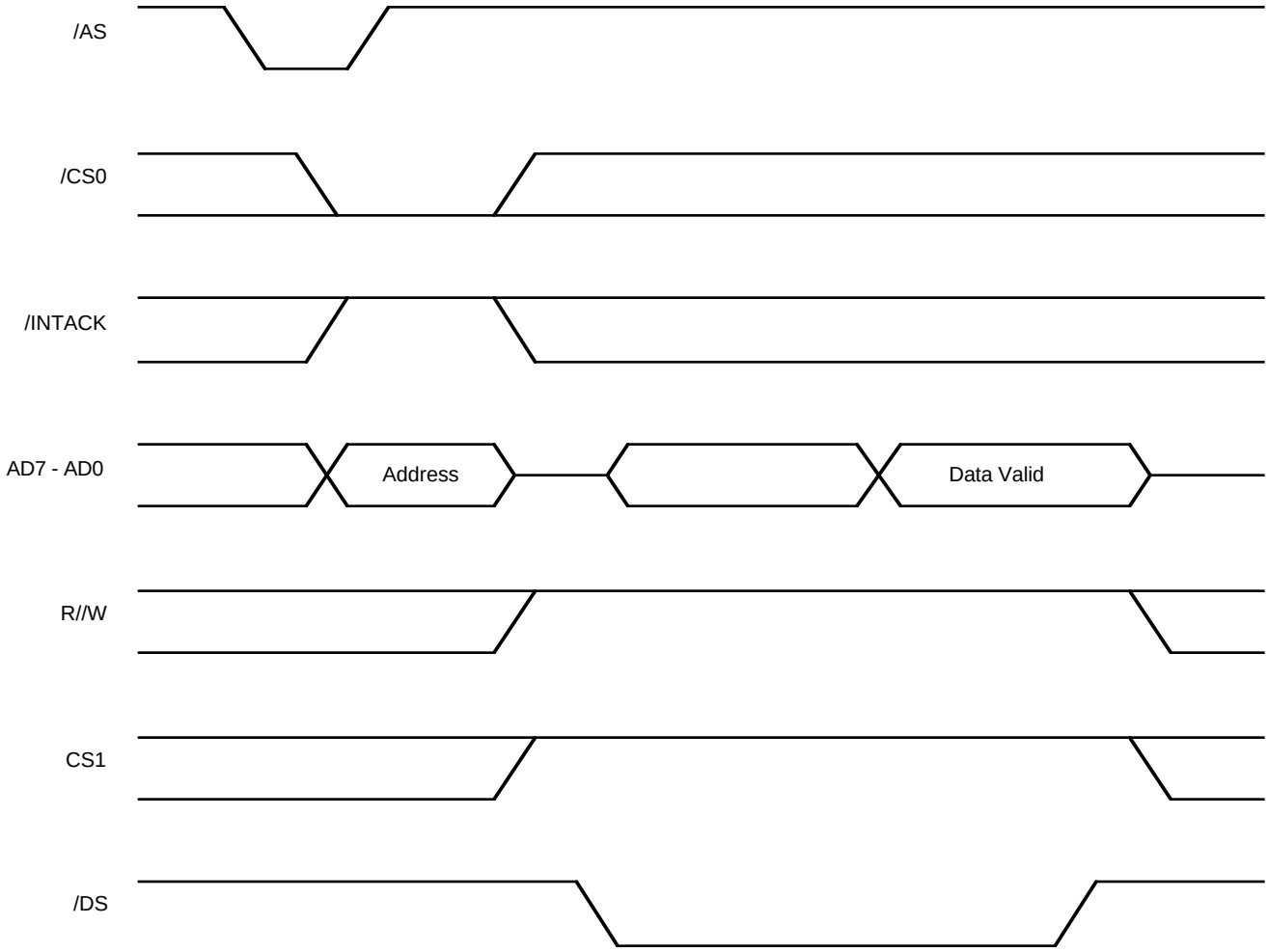



**Figure 2-1. Z80X30 Read Cycle**

### 2.2.2 Z80X30 Write Cycle Timing

The write cycle timing for the Z80X30 is shown in Figure 2-2. The register address on AD7-AD0, as well as the state of /CS0 and /INTACK, are latched by the rising edge of /AS. R//W must be Low when /DS falls to indicate a write cycle. The leading edge of the coincidence of CS1 High and /DS Low latches the write data on AD7-AD0, as well as the state of R//W.

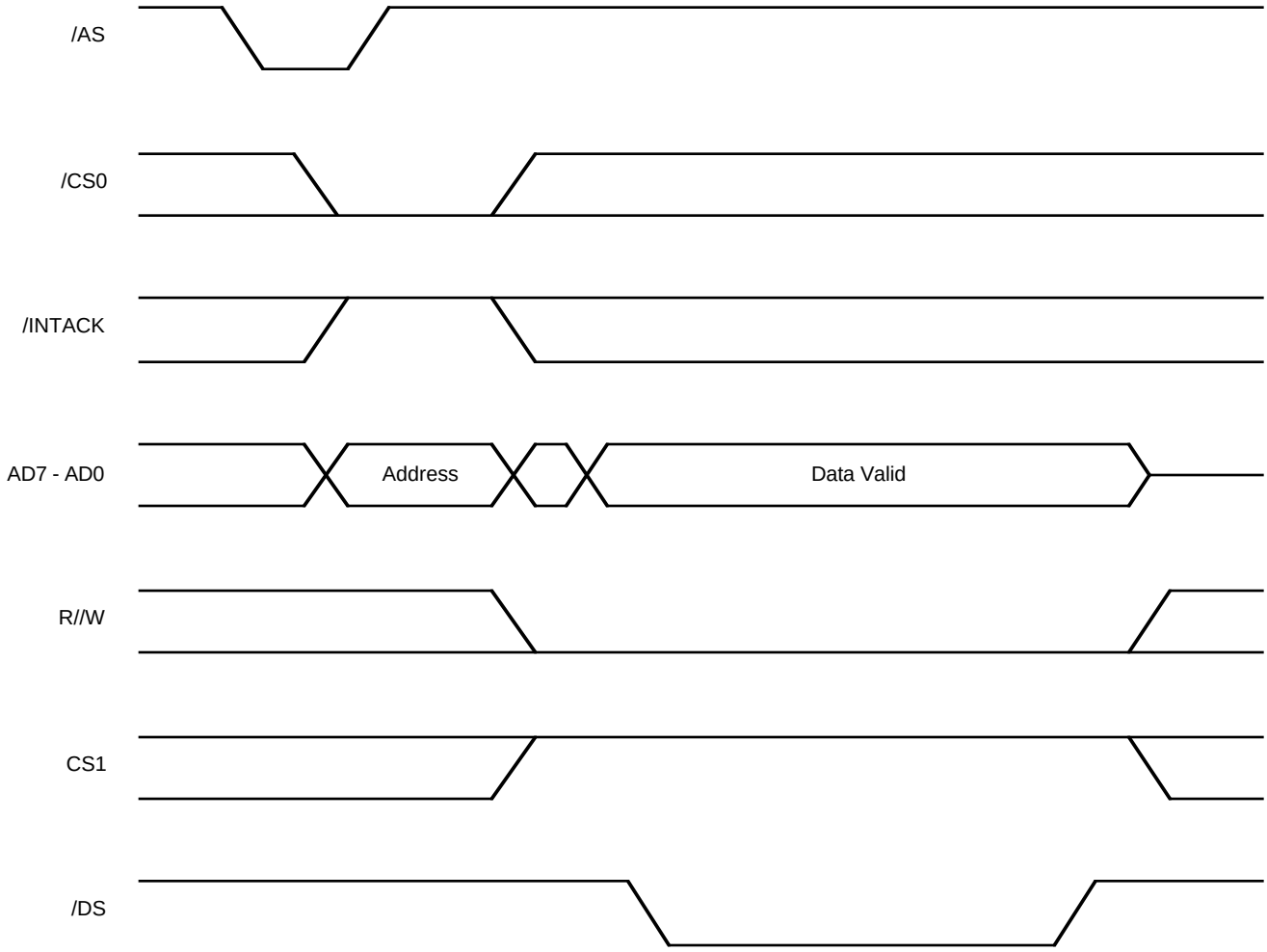



**Figure 2-2. Z80X30 Write Cycle**

## 2.2 Z80X30 INTERFACE TIMING (Continued)

### 2.2.3 Z80X30 Interrupt Acknowledge Cycle Timing

The interrupt acknowledge cycle timing for the Z80X30 is shown in Figure 2-3. The address on AD7-AD0 and the state of /CS0 and /INTACK are latched by the rising edge of /AS. However, if /INTACK is Low, the address, /CS0, CS1 and R//W are ignored for the duration of the interrupt acknowledge cycle.

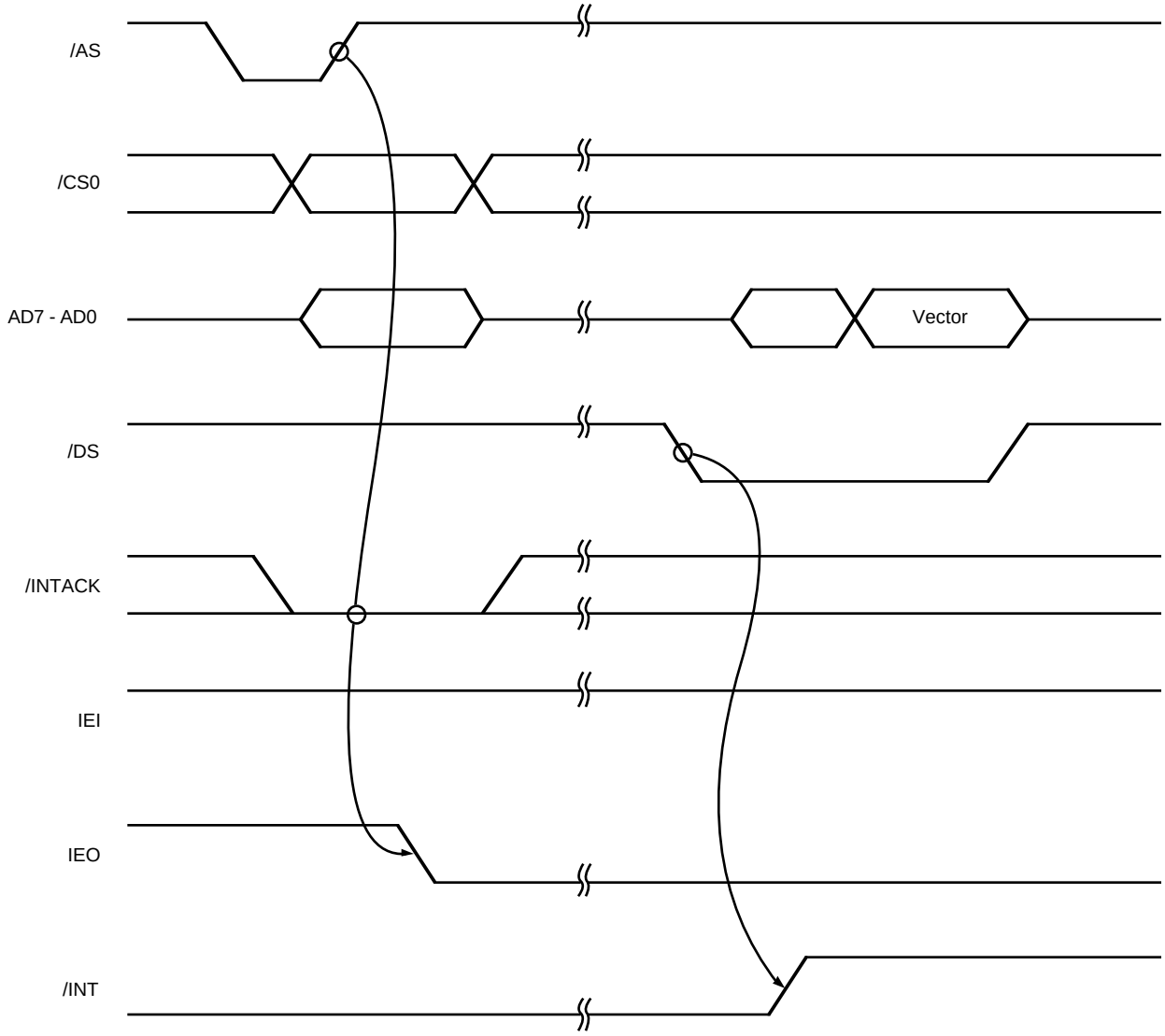



**Figure 2-3. Z80X30 Interrupt Acknowledge Cycle**

The Z80X30 samples the state of /INTACK on the rising edge of /AS, and AC parameters #7 and #8 specify the set-up and hold-time requirements. Between the rising edge of /AS and the falling edge of /DS, the internal and external daisy chains settle (AC parameter #29). A system with no external daisy chain should provide the time specified in spec #29 to settle the interrupt daisy-chain priority internal to the SCC. Systems using an external daisy chain should refer to Note 5 referenced in the Z80X30 Read/Write & Interrupt Acknowledge Timing for the time required to settle the daisy chain.

**Note:** /INTACK is sampled on the rising edge of /AS. If it does not meet the setup time to the first rising edge of /AS of the interrupt acknowledge cycle, it is latched on the next rising edge of /AS. Therefore, if /INTACK is asynchronous to /AS, it may be necessary to add a PCLK cycle to the calculation for /INTACK to /RD delay time.

If there is an interrupt pending in the SCC, and IEI is High when /DS falls, the acknowledge cycle was intended for the SCC. This being the case, the Z80X30 sets the Interrupt-Under-Service (IUS) latch for the highest priority pending interrupt, as well as placing an interrupt vector on AD7-AD0. The placing of a vector on the bus can be disabled by setting WR9, D1=1. The /INT pin also goes inactive in response to the falling edge of /DS. Note that there should be only one /DS per acknowledge cycle. Another important fact is that the IP bits in the Z80X30 are updated by /AS, which may delay interrupt requests if the processor does not supply /AS strobes during the time between accesses of the Z80X30.

### 2.2.4 Z80X30 Register Access

The registers in the Z80X30 are addressed via the address on AD7-AD0 and are latched by the rising edge of /AS. The Shift Right/Shift Left bit in the Channel B WR0 controls which bits are decoded to form the register address. It is placed in this register to simplify programming when the current state of the Shift Right/Shift Left bit is not known.

A hardware reset forces Shift Left mode where the address is decoded from AD5-AD1. In Shift Right mode, the address is decoded from AD4-AD0. The Shift Right/Shift Left bit is written via a command to make the software writing to WR0 independent of the state of the Shift Right/Shift Left bit.

While in the Shift Left mode, the register address is placed on AD4-AD1 and the Channel Select bit, A/B, is decoded from AD5. The register map for this case is shown in Table 2-1. In Shift Right mode, the register address is again placed on AD4-AD1 but the channel select A/B is decoded from AD0. The register map for this case is shown in Table 2-2.

Because the Z80X30 does not contain 16 read registers, the decoding of the read registers is not complete; this is indicated in Table 2-1 and Table 2-2 by parentheses around the register name. These addresses may also be used to access the read registers. Also, note that the Z80X30 contains only one WR2 and WR9; these registers may be written from either channel.

Shift Left Mode is used when Channel A and B are to be programmed differently. This allows the software to sequence through the registers of one channel at a time. The Shift Right Mode is used when the channels are programmed the same. By incrementing the address, the user can program the same data value into both the Channel A and Channel B register.

## 2.2 Z80X30 INTERFACE TIMING (Continued)

**Table 2-1. Z80X30 Register Map (Shift Left Mode)**

| AD5 | AD4 | AD3 | AD2 | AD1 | WRITE | READ 8030 80C30/230* WR15 D2 = 0 | 80C30/230 WR15 D2=1 | 80230 WR15 D2=1 WR7' D6=1 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | WR0B | RR0B | RR0B | RR0B |
| 0 | 0 | 0 | 0 | 1 | WR1B | RR1B | RR1B | RR1B |
| 0 | 0 | 0 | 1 | 0 | WR2 | RR2B | RR2B | RR2B |
| 0 | 0 | 0 | 1 | 1 | WR3B | RR3B | RR3B | RR3B |
| 0 | 0 | 1 | 0 | 0 | WR4B | (RR0B) | (RR0B) | (WR4B) |
| 0 | 0 | 1 | 0 | 1 | WR5B | (RR1B) | (RR1B) | (WR5B) |
| 0 | 0 | 1 | 1 | 0 | WR6B | (RR2B) | RR6B | RR6B |
| 0 | 0 | 1 | 1 | 1 | WR7B | (RR3B) | RR7B | RR7B |
| 0 | 1 | 0 | 0 | 0 | WR8B | RR8B | RR8B | RR8B |
| 0 | 1 | 0 | 0 | 1 | WR9 | (RR13B) | (RR13B) | (WR3B) |
| 0 | 1 | 0 | 1 | 0 | WR10B | RR10B | RR10B | RR10B |
| 0 | 1 | 0 | 1 | 1 | WR11B | (RR15B) | (RR15B) | (WR10B) |
| 0 | 1 | 1 | 0 | 0 | WR12B | RR12B | RR12B | RR12B |
| 0 | 1 | 1 | 0 | 1 | WR13B | RR13B | RR13B | RR13B |
| 0 | 1 | 1 | 1 | 0 | WR14B | RR14B | RR14B | (WR7'B) |
| 0 | 1 | 1 | 1 | 1 | WR15B | RR15B | RR15B | RR15B |
| 1 | 0 | 0 | 0 | 0 | WR0A | RR0A | RR0A | RR0A |
| 1 | 0 | 0 | 0 | 1 | WR1A | RR1A | RR1A | RR1A |
| 1 | 0 | 0 | 1 | 0 | WR2 | RR2A | RR2A | RR2A |
| 1 | 0 | 0 | 1 | 1 | WR3A | RR3A | RR3A | RR3A |
| 1 | 0 | 1 | 0 | 0 | WR4A | (RR0A) | (RR0A) | (WR4A) |
| 1 | 0 | 1 | 0 | 1 | WR5A | (RR1A) | (RR1A) | (WR5A) |
| 1 | 0 | 1 | 1 | 0 | WR6A | (RR2A) | RR6A | RR6A |
| 1 | 0 | 1 | 1 | 1 | WR7A | (RR3A) | RR7A | RR7A |
| 1 | 1 | 0 | 0 | 0 | WR8A | RR8A | RR8A | RR8A |
| 1 | 1 | 0 | 0 | 1 | WR9 | (RR13A) | (RR13A) | (WR3A) |
| 1 | 1 | 0 | 1 | 0 | WR10A | RR10A | RR10A | RR10A |
| 1 | 1 | 0 | 1 | 1 | WR11A | (RR15A) | (RR15A) | (WR10A) |
| 1 | 1 | 1 | 0 | 0 | WR12A | RR12A | RR12A | RR12A |
| 1 | 1 | 1 | 0 | 1 | WR13A | RR13A | RR13A | RR13A |
| 1 | 1 | 1 | 1 | 0 | WR14A | RR14A | RR14A | (WR7'A) |
| 1 | 1 | 1 | 1 | 1 | WR15A | RR15A | RR15A | RR15A |

**Notes:**
The register names in ( ) are the values read out from that register location.
WR15, bit D2 enables status FIFO function (not available on NMOS).
WR7' bit D6 enables extend read function (only on ESCC).
* Includes 80C30/230 when WR15 D2=0.

**Table 2-2. Z80X30 Register Map (Shift Right Mode)**

| AD4 | AD3 | AD2 | AD1 | AD0 | WRITE | READ 8030 80C30/230* WR15 D2 = 0 | 80C30/230 WR15 D2=1 | 80230 WR15 D2=1 WR7' D6=1 |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | WR0B | RR0B | RR0B | RR0B |
| 0 | 0 | 0 | 0 | 1 | WR0A | RR0A | RR0A | RR0A |
| 0 | 0 | 0 | 1 | 0 | WR1B | RR1B | RR1B | RR1B |
| 0 | 0 | 0 | 1 | 1 | WR1A | RR1A | RR1A | RR1A |
| 0 | 0 | 1 | 0 | 0 | WR2 | RR2B | RR2B | RR2B |
| 0 | 0 | 1 | 0 | 1 | WR2 | RR2A | RR2A | RR2A |
| 0 | 0 | 1 | 1 | 0 | WR3B | RR3B | RR3B | RR3B |
| 0 | 0 | 1 | 1 | 1 | WR3A | RR3A | RR3A | RR3A |
| 0 | 1 | 0 | 0 | 0 | WR4B | (RR0B) | (RR0B) | (WR4B) |
| 0 | 1 | 0 | 0 | 1 | WR4A | (RR0A) | (RR0A) | (WR4A) |
| 0 | 1 | 0 | 1 | 0 | WR5B | (RR1B) | (RR1B) | (WR5B) |
| 0 | 1 | 0 | 1 | 1 | WR5A | (RR1A) | (RR1A) | (WR5A) |
| 0 | 1 | 1 | 0 | 0 | WR6B | (RR2B) | RR6B | RR6B |
| 0 | 1 | 1 | 0 | 1 | WR6A | (RR2A) | RR6A | RR6A |
| 0 | 1 | 1 | 1 | 0 | WR7B | (RR3B) | RR7B | RR7B |
| 0 | 1 | 1 | 1 | 1 | WR7A | (RR3A) | RR7A | RR7A |
| 1 | 0 | 0 | 0 | 0 | WR8B | RR8B | RR8B | RR8B |
| 1 | 0 | 0 | 0 | 1 | WR8A | RR8A | RR8A | RR8A |
| 1 | 0 | 0 | 1 | 0 | WR9 | (RR13B) | (RR13B) | (WR3B) |
| 1 | 0 | 0 | 1 | 1 | WR9 | (RR13A) | (RR13A) | (WR3A) |
| 1 | 0 | 1 | 0 | 0 | WR10B | RR10B | RR10B | RR10B |
| 1 | 0 | 1 | 0 | 1 | WR10A | RR10A | RR10A | RR10A |
| 1 | 0 | 1 | 1 | 0 | WR11B | (RR15B) | (RR15B) | (WR10B) |
| 1 | 0 | 1 | 1 | 1 | WR11A | (RR15A) | (RR15A) | (WR10A) |
| 1 | 1 | 0 | 0 | 0 | WR12B | RR12B | RR12B | RR12B |
| 1 | 1 | 0 | 0 | 1 | WR12A | RR12A | RR12A | RR12A |
| 1 | 1 | 0 | 1 | 0 | WR13B | RR13B | RR13B | RR13B |
| 1 | 1 | 0 | 1 | 1 | WR13A | RR13A | RR13A | RR13A |
| 1 | 1 | 1 | 0 | 0 | WR14B | RR14B | RR14B | (WR7'B) |
| 1 | 1 | 1 | 0 | 1 | WR14A | RR14A | RR14A | (WR7'A) |
| 1 | 1 | 1 | 1 | 0 | WR15B | RR15B | RR15B | RR15B |
| 1 | 1 | 1 | 1 | 1 | WR15A | RR15A | RR15A | RR15A |

**Notes:**
The register names in ( ) are the values read out from that register location.
WR15 bit D2 enables status FIFO function (not available on NMOS).
WR7' bit D6 enables extend read function (only on ESCC).
* Includes 80C30/230 when WR15 D2=0.

## 2.2 Z80X30 INTERFACE TIMING (Continued)

### 2.2.5 Z80C30 Register Enhancement

The Z80C30 has an enhancement to the NMOS Z8030 register set, which is the addition of a 10x19 SDLC Frame Status FIFO. When WR15 bit D2=1, the SDLC Frame Status FIFO is enabled, and it changes the functionality of RR6 and RR7. See Section 4.4.3 for more details on this feature.

### 2.2.6 Z80230 Register Enhancements

In addition to the Z80C30 enhancements, the 80230 has several enhancements to the SCC register set. These include the addition of Write Register 7 Prime (WR7'), and the ability to read registers that are read only in the 8030.

Write Register 7' is addressed by setting WR15 bit, D0=1 and then addressing WR7. Figure 2-4 shows the register bit location of the six features enabled through this register. All writes to address seven are to WR7' when WR15, D0=1. Refer to Chapter 5 for detailed information on WR7'.

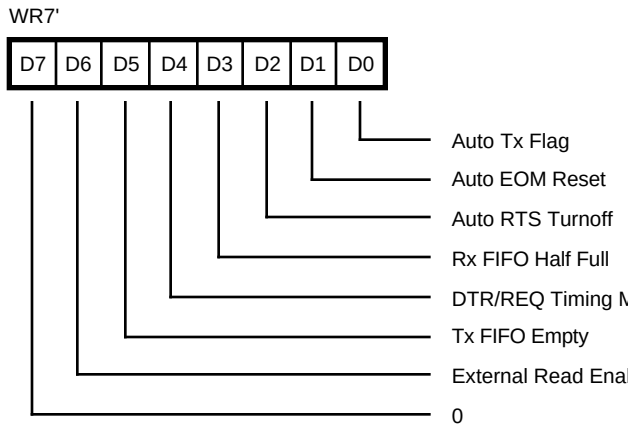



**Figure 2-4. Write Register 7 Prime (WR7')**

WR7' bit D6=1, enables the extended read register capability. This allows the user to read the contents of WR3, WR4, WR5, WR7' and WR10 by reading RR9, RR4, RR5, RR14 and RR11, respectively. When WR7' D6=0, these write registers are write only.

Table 2-3 shows what functions are enabled for the various combinations of register bit enables. See Table 2-1 (Shift Left) and Table 2-2 (Shift Right) for the register address map with the SDLC FIFO enabled only and the map with both the extended read and SDLC FIFO features enabled.

**Table 2-3. Z80230 SDLC/HDLC Enhancement Options**

| WR15 | | WR7' | |
|---|---|---|---|
| **Bit D2** | **Bit D0** | **Bit D6** | **Functions Enabled** |
| 0 | 1 | 0 | WR7' enabled only |
| 0 | 1 | 1 | WR7' with extended read enabled |
| 1 | 0 | X | 10x19 SDLC FIFO enhancement enabled only |
| 1 | 1 | 0 | 10x19 SDLC FIFO and WR7' |
| 1 | 1 | 1 | 10x19 SDLC FIFO and WR7' with extended read enabled |

### 2.2.7 Z80X30 Reset

The Z80X30 may be reset by either a hardware or software reset. Hardware reset occurs when /AS and /DS are both Low at the same time, which is normally an illegal condition. As long as both /AS and /DS are Low, the Z80X30 recognizes the reset condition. However, once this condition is removed, the reset condition is asserted internally for an additional four to five PCLK cycles. During this time, any attempt to access is ignored.

The Z80X30 has three software resets that are encoded into two command bits in WR9. There are two channel resets, which only affect one channel in the device and some bits of the write registers. The command forces the same result as the hardware reset, the Z80X30 stretches the reset signal an additional four to five PCLK cycles beyond the ordinary valid access recovery time. The bits in WR9 may be written at the same time as the reset command because these bits are affected only by a hardware reset. The reset values of the various registers are shown in Table 2-4.

**Table 2-4. Z80X30 Register Reset Values**

| | Hardware RESET | | | | | | | | Channel RESET | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| WR0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| WR1 | 0 | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | X | 0 | 0 | X | 0 | 0 |
| WR2 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR3 | X | X | X | X | X | X | X | 0 | X | X | X | X | X | X | X | 0 |
| WR4 | X | X | X | X | X | 1 | X | X | X | X | X | X | X | 1 | X | X |
| WR5 | 0 | X | X | 0 | 0 | 0 | 0 | X | 0 | X | X | 0 | 0 | 0 | 0 | X |
| WR6 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR7 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR7'* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| WR9 | 1 | 1 | 0 | 0 | 0 | 0 | X | X | X | X | 0 | X | X | X | X | X |
| WR10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | 0 | 0 | 0 | 0 | 0 |
| WR11 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X | X | X | X | X | X | X | X |
| WR12 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR13 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR14 | X | X | 1 | 1 | 0 | 0 | 0 | 0 | X | X | 1 | 0 | 0 | 0 | X | X |
| WR15 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| RR0 | X | 1 | X | X | X | 1 | 0 | 0 | X | 1 | X | X | X | 1 | 0 | 0 |
| RR1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X |
| RR3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RR10 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 |

**Notes:**
*'WR7' is available only on the Z80230.

## 2.3 Z85X30 INTERFACE TIMING

Two control signals, /RD and /WR, are used by the Z85X30 to time bus transactions. In addition, four other control signals, /CE, D//C, A//B and /INTACK, are used to control the type of bus transaction that occurs. A bus transaction starts when the addresses on D//C and A//B are asserted before /RD or /WR fall (AC Spec #6 and #8). The coincidence of /CE and /RD or /CE and /WR latches the state of D//C and A//B and starts the internal operation. The /INTACK signal must have been previously sampled High by a rising edge of PCLK for a read or write cycle to occur. In addition to sampling /INTACK, PCLK is used by the interrupt section to set the IP bits.

The Z85X30 generates internal control signals in response to a register access. Since /RD and /WR have no phase relationship with PCLK, the circuitry generating these internal control signals provides time for metastable conditions to disappear. This results in a recovery time related to PCLK.

This recovery time applies only between transactions involving the Z85X30, and any intervening transactions are ignored. This recovery time is four PCLK cycles (AC Spec #49), measured from the falling edge of /RD or /WR in the case of a read or write of any register.

### 2.3.1 Z85X30 Read Cycle Timing

The read cycle timing for the Z85X30 is shown in Figure 2-5. The address on A//B and D//C is latched by the coincidence of /RD and /CE active. /CE must remain Low and /INTACK must remain High throughout the cycle. The Z85X30 bus drivers are enabled while /CE and /RD are both Low. A read with D//C High does not disturb the state of the pointers and a read cycle with D//C Low resets the pointers to zero after the internal operation is complete

.

**Figure 2-5. Z85X30 Read Cycle Timing**

### 2.3.2 Z85X30 Write Cycle Timing

The write cycle timing for the Z85X30 is shown in Figure 2-6. The address on A//B and D//C, as well as the data on D7-D0, is latched by the coincidence of /WR and /CE active. /CE must remain Low and /INTACK must remain High throughout the cycle. A write cycle with D//C High does not disturb the state of the pointers and a write cycle with D//C Low resets the pointers to zero after the internal operation is complete.

Historically, the NMOS/CMOS version latched the data bus on the falling edge of /WR. However, many CPUs do not guarantee that the data bus is valid at the time when the /WR pin goes low, so the data bus timing was modified to allow a maximum delay from the falling edge of /WR to the latching of the data bus. On the Z85230, the AC Timing parameter #29 TsDW(WR), Write Data to /WR falling minimum, has been changed to: /WR falling to Write Data Valid maximum. Refer to the AC Timing Characteristic section of the Z85230 Product Specification for more information regarding this change.

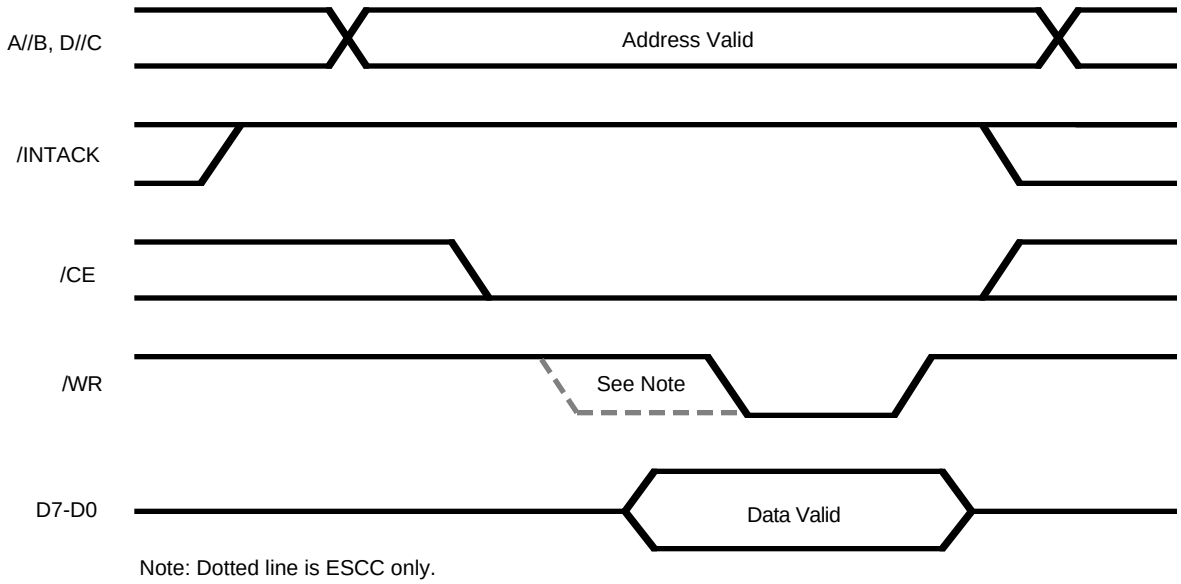



**Figure 2-6. Z85X30 Write Cycle Timing**

### 2.3.3 Z85X30 Interrupt Acknowledge Cycle Timing

The interrupt acknowledge cycle timing for the Z85X30 is shown in Figure 2-7. The state of /INTACK is latched by the rising edge of PCLK (AC Spec #10). While /INTACK is Low, the state of A//B, /CE, D//C, and /WR are ignored.

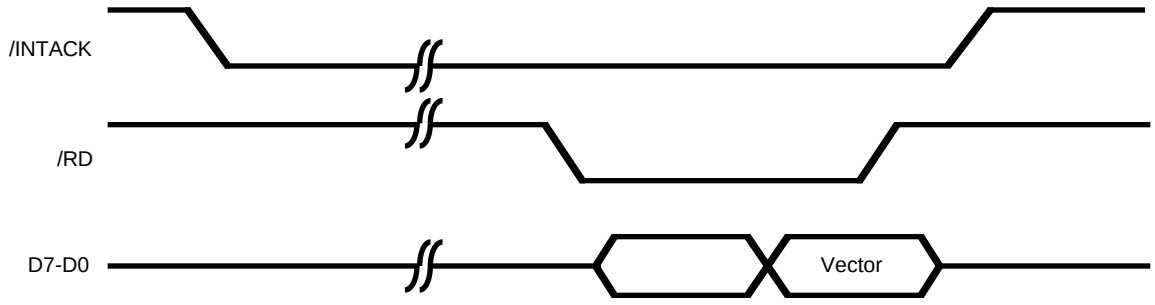



**Figure 2-7. Z85X30 Interrupt Acknowledge Cycle Timing**

## 2.3 Z85X30 INTERFACE TIMING (Continued)

Between the time /INTACK is first sampled Low and the time /RD falls, the internal and external IEI/IEO daisy chain settles (AC parameter #38 TdIAI(RD) Note 5). A system with no external daisy chain must provide the time specified in AC Spec #38 to settle the interrupt daisy chain priority internal to the SCC. Systems using the external IEI/IEO daisy chain should refer to Note 5 referenced in the Z85X30 Read/Write and Interrupt Acknowledge Timing for the time required to settle the daisy chain.

**Note:** /INTACK is sampled on the rising edge of PCLK. If it does not meet the setup time to the first rising edge of PCLK of the interrupt acknowledge cycle, it is latched on the next rising edge of PCLK. Therefore, if /INTACK is asynchronous to PCLK, it may be necessary to add a PCLK cycle to the calculation for /INTACK to /RD delay time.

If there is an interrupt pending in the Z85X30, and IEI is High when /RD falls, the interrupt acknowledge cycle was intended for the Z85X30. In this case, the Z85X30 sets the appropriate Interrupt-Under-Service latch, and places an interrupt vector on D7-D0.

If the falling edge of /RD sets an IUS bit in the Z85X30, the /INT pin goes inactive in response to the falling edge. Note that there should be only one /RD per acknowledge cycle.

**Note 1:** The IP bits in the Z85X30 are updated by PCLK. However, when the register pointer is pointing to RR2 and RR3, the IP bits are prevented from changing. This prevents data changing during a read, but will delay interrupt requests if the pointers are left pointing at these registers.

**Note 2:** The SCC should only receive one INTACK signal per acknowledge cycle. Therefore, if the CPU generates more than one (as is common for the 80X86 family), an external circuit should be used to convert this into a single pulse or does not use Interrupt Acknowledge.

### 2.3.4 Z85X30 Register Access

The registers in the Z85X30 are accessed in a two step process, using a Register Pointer to perform the addressing. To access a particular register, the pointer bits are set by writing to WR0. The pointer bits may be written in either channel because only one set exists in the Z85X30. After the pointer bits are set, the next read or write cycle of the Z85X30 having D//C Low will access the desired register. At the conclusion of this read or write cycle the pointer bits are reset to 0s, so that the next control write is to the pointers in WR0.

A read to RR8 (the receive data FIFO) or a write to WR8 (the transmit data FIFO) is either done in this fashion or by accessing the Z85X30 having D//C pin High. A read or write with D//C High accesses the data registers directly, and independently of the state of the pointer bits. This allows single-cycle access to the data registers and does not disturb the pointer bits.

The fact that the pointer bits are reset to 0, unless explicitly set otherwise, means that WR0 and RR0 may also be accessed in a single cycle. That is, it is not necessary to write the pointer bits with 0 before accessing WR0 or RR0.

There are three pointer bits in WR0, and these allow access to the registers with addresses 7 through 0. Note that a command may be written to WR0 at the same time that the pointer bits are written. To access the registers with addresses 15 through 8, the Point High command must accompany the pointer bits. This precludes concurrently issuing a command when pointing to these registers.

The register map for the Z85X30 is shown in Table 2-5. If, for some reason, the state of the pointer bits is unknown they may be reset to 0 by performing a read cycle with the D//C pin held Low. Once the pointer bits have been set, the desired channel is selected by the state of the A//B pin during the actual read or write of the desired register.

**Table 2-5. Z85X30 Register Map**

| A//B | PNT2 | PNT1 | PNT0 | WRITE | Read 8530 85C30/230 WR15 D2 = 0 | 85C30/230 WR15 D2=1 | WR15 D2=1 WR7' D6=1 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | WR0B | RR0B | RR0B | RR0B |
| 0 | 0 | 0 | 1 | WR1B | RR1B | RR1B | RR1B |
| 0 | 0 | 1 | 0 | WR2 | RR2B | RR2B | RR2B |
| 0 | 0 | 1 | 1 | WR3B | RR3B | RR3B | RR3B |
| 0 | 1 | 0 | 0 | WR4B | (RR0B) | (RR0B) | (WR4B) |
| 0 | 1 | 0 | 1 | WR5B | (RR1B) | (RR1B) | (WR5B) |
| 0 | 1 | 1 | 0 | WR6B | (RR2B) | RR6B | RR6B |
| 0 | 1 | 1 | 1 | WR7B | (RR3B) | RR7B | RR7B |
| 1 | 0 | 0 | 0 | WR0A | RR0A | RR0A | RR0A |
| 1 | 0 | 0 | 1 | WR1A | RR1A | RR1A | RR1A |
| 1 | 0 | 1 | 0 | WR2 | RR2A | RR2A | RR2A |
| 1 | 0 | 1 | 1 | WR3A | RR3A | RR3A | RR3A |
| 1 | 1 | 0 | 0 | WR4A | (RR0A) | (RR0A) | (WR4A) |
| 1 | 1 | 0 | 1 | WR5A | (RR1A) | (RR1A) | (WR5A) |
| 1 | 1 | 1 | 0 | WR6A | (RR2A) | RR6A | RR6A |
| 1 | 1 | 1 | 1 | WR7A | (RR3A) | RR7A | RR7A |
| **With Point High Command** | | | | | | | |
| 0 | 0 | 0 | 0 | WR8B | RR8B | RR8B | RR8B |
| 0 | 0 | 0 | 1 | WR9 | (RR13B) | (RR13B) | (WR3B) |
| 0 | 0 | 1 | 0 | WR10B | RR10B | RR10B | RR10B |
| 0 | 0 | 1 | 1 | WR11B | (RR15B) | (RR15B) | (WR10B) |
| 0 | 1 | 0 | 0 | WR12B | RR12B | RR12B | RR12B |
| 0 | 1 | 0 | 1 | WR13B | RR13B | RR13B | RR13B |
| 0 | 1 | 1 | 0 | WR14B | RR14B | RR14B | (WR7'B) |
| 0 | 1 | 1 | 1 | WR15B | RR15B | RR15B | RR15B |
| 1 | 0 | 0 | 0 | WR8A | RR8A | RR8A | RR8A |
| 1 | 0 | 0 | 1 | WR9 | (RR13A) | (RR13A) | (WR3A) |
| 1 | 0 | 1 | 0 | WR10A | RR10A | RR10A | RR10A |
| 1 | 0 | 1 | 1 | WR11A | (RR15A) | (RR15A) | (WR10A) |
| 1 | 1 | 0 | 0 | WR12A | RR12A | RR12A | RR12A |
| 1 | 1 | 0 | 1 | WR13A | RR13A | RR13A | RR13A |
| 1 | 1 | 1 | 0 | WR14A | RR14A | RR14A | (WR7'A) |
| 1 | 1 | 1 | 1 | WR15A | RR15A | RR15A | RR15A |

**Notes:**
WR15 bit D2 enables status FIFO function. (Not available on NMOS)
WR7' bit D6 enables extend read function. (Only on ESCC and 85C30)

## 2.3 Z85X30 INTERFACE TIMING (Continued)

### 2.3.5 Z85C30 Register Enhancement

The Z85C30 has an enhancement to the NMOS Z8530 register set, which is the addition of a 10x19 SDLC Frame Status FIFO. When WR15 bit D2=1, the SDLC Frame Status FIFO is enabled, and it changes the functionality of RR6 and RR7. See Section 4.4.3 for more details on this feature.

### 2.3.6 Z85C30/Z85230 Register Enhancements

In addition to the enhancements mentioned in 2.3.5, the 85C30/85230 provides several enhancements to the SCC register set. These include the addition of Write Register 7 Prime (WR7'), the ability to read registers that are write-only in the SCC.

Write Register 7' is addressed by setting WR15, D0=1 and then addressing WR7. Figure 2-8 shows the register bit location of the six features enabled through this register for the 85230, while Figure 2-7 shows the register bit location for the 85C30. Note that the difference between the two WR7' registers for the 85230 and the 85C30 is bit D5 and bit D4. All writes to address seven are to WR7' when WR15 D0=1. Refer to Chapter 5 for detailed information on WR7'.

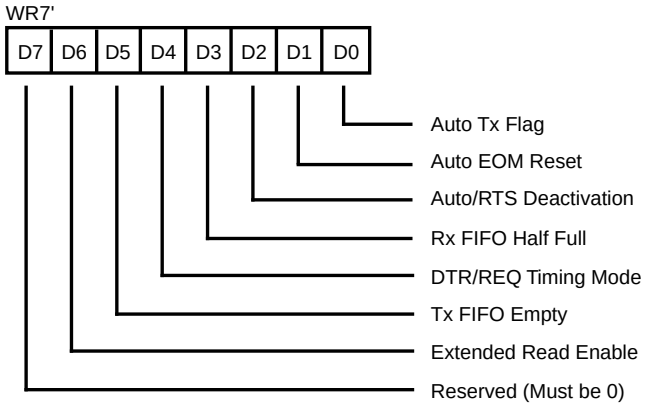



**Figure 2-8a. Write Register 7 Prime (WR7') for the 85230**

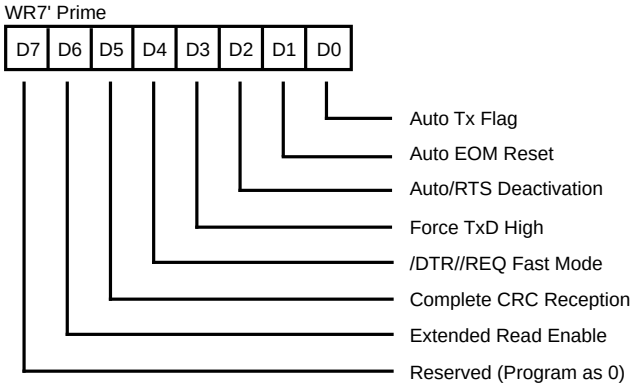



**Figure 2-8b. Write Register 7 Prime for the 85C30**

Setting WR7' bit D6=1 enables the extended read register capability. This allows the user to read the contents of WR3, WR4, WR5, WR7' and WR10 by reading RR9, RR4, RR5, RR14 and RR11, respectively. When WR7' D6=0, these write registers are write-only.

Table 2-6 shows what functions are enabled for the various combinations of register bit enables. See Table 2-5 for the register address map with only the SDLC FIFO enabled and with both the extended read and SDLC FIFO features enabled.

**Table 2-6. Z85C30/Z85230 Register Enhancement Options**

| WR15 | | WR7' | |
|---|---|---|---|
| **Bit D2** | **Bit D0** | **Bit D6** | **Functions Enabled** |
| 0 | 1 | 0 | WR7' enabled only |
| 0 | 1 | 1 | WR7' with extended read enabled |
| 1 | 0 | X | 10x19 SDLC FIFO enhancement enabled only |
| 1 | 1 | 0 | 10x19 SDLC FIFO and WR7' |
| 1 | 1 | 1 | 10x19 SDLC FIFO and WR7' with extended read enabled |

### 2.3.7 Z85X30 Reset

The Z85X30 may be reset by either a hardware or software reset. Hardware reset occurs when /WR and /RD are both Low at the same time, which is normally an illegal condition. As long as both /WR and /RD are Low, the Z85X30 recognizes the reset condition. However, once this condition is removed, the reset condition is asserted internally for an additional four to five PCLK cycles. During this time any attempt to access is ignored.

The Z85X30 has three software resets that are encoded into the command bits in WR9. There are two channel resets which only affect one channel in the device and some bits of the write registers. The command forces the same result as the hardware reset, the Z85X30 stretches the reset signal an additional four to five PCLK cycles beyond the ordinary valid access recovery time. The bits in WR9 may be written at the same time as the reset command because these bits are affected only by a hardware reset. The reset values of the various registers are shown in Table 2-7.

**Table 2-7. Z85X30 Register Reset Value**

| | Hardware RESET | | | | | | | | Channel RESET | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** | **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| WR0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| WR1 | 0 | 0 | X | 0 | 0 | X | 0 | 0 | 0 | 0 | X | 0 | 0 | X | 0 | 0 |
| WR2 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR3 | X | X | X | X | X | X | X | 0 | X | X | X | X | X | X | X | 0 |
| WR4 | X | X | X | X | X | 1 | X | X | X | X | X | X | X | 1 | X | X |
| WR5 | 0 | X | X | 0 | 0 | 0 | 0 | X | 0 | X | X | 0 | 0 | 0 | 0 | X |
| WR6 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR7 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR7'* | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| WR9 | 1 | 1 | 0 | 0 | 0 | 0 | X | X | X | X | 0 | X | X | X | X | X |
| WR10 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | X | 0 | 0 | 0 | 0 | 0 |
| WR11 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | X | X | X | X | X | X | X | X |
| WR12 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR13 | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X | X |
| WR14 | X | X | 1 | 1 | 0 | 0 | 0 | 0 | X | X | 1 | 0 | 0 | 0 | X | X |
| WR15 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| RR0 | X | 1 | X | X | X | 1 | 0 | 0 | X | 1 | X | X | X | 1 | 0 | 0 |
| RR1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X | 0 | 0 | 0 | 0 | 0 | 1 | 1 | X |
| RR3 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| RR10 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 | 0 | X | 0 | 0 | 0 | 0 | 0 | 0 |

**Notes:**
*WR7' is only available on the 85C30 and the ESCC.

## 2.4 INTERFACE PROGRAMMING

The following subsections explain and illustrate all areas of interface programming.

### 2.4.1 I/O Programming Introduction

The SCC can work with three basic forms of I/O operations: polling, interrupts, and block transfer. All three I/O types involve register manipulation during initialization and data transfer. However, the interrupt mode also incorporates Z-Bus interrupt protocol for a fast and efficient data transfer.

Regardless of the version of the SCC, all communication modes can use a choice of polling, interrupt and block transfer. These modes are selected by the user to determine the proper hardware and software required to supply data at the rate required.

**Note to ESCC Users:** Those familiar with the NMOS/CMOS version will find the ESCC I/O operations very similar but should note the following differences: the addition of software acknowledge (which is available in the current version of the CMOS SCC, but not in NMOS); the /DTR//REQ pin can be programmed to be deasserted faster; and the programmability of the data interrupts to the FIFO fill level.

## 2.4 INTERFACE PROGRAMMING (Continued)

### 2.4.2 Polling

This is the simplest mode to implement. The software must poll the SCC to determine when data is to be input or output from the SCC. In this mode, MIE (WR9, bit 3), and Wait/DMA Request Enable (WR1, bit 7) are both reset to 0 to disable any interrupt or DMA requests. The software must then poll RR0 to determine the status of the receive buffer, transmit buffer and external status.

During a polling sequence, the status of Read Register 0 is examined in each channel. This register indicates whether or not a receive or transmit data transfer is needed and whether or not any special conditions are present, e.g., errors.

This method of I/O transfer avoids interrupts and, consequently, all interrupt functions should be disabled. With no interrupts enabled, this mode of operation must initiate a read cycle of Read Register 0 to detect an incoming character before jumping to a data handler routine.

### 2.4.3 Interrupts

Each of the SCC's two channels contain three sources of interrupts, making a total of six interrupt sources. These three sources of interrupts are: 1) Receiver, 2) Transmitter, and 3) External/Status conditions. In addition, there are several conditions that may cause these interrupts. Figure 2-9 shows the different conditions for each interrupt source and each is enabled under program control. Channel A has a higher priority than Channel B with Receive, Transmit, and External/Status Interrupts prioritized, respectively, within each channel as shown in Table 2-8. The SCC internally updates the interrupt status on every PCLK cycle in the Z85X30 and on /AS in the Z80X30.

**Table 2-8. Interrupt Source Priority**

| | |
|---|---|
| Receive Channel A | Highest |
| Transmit Channel A | |
| External/Status Channel A | |
| Receive Channel B | |
| Transmit Channel B | |
| External/Status Channel B | Lowest |

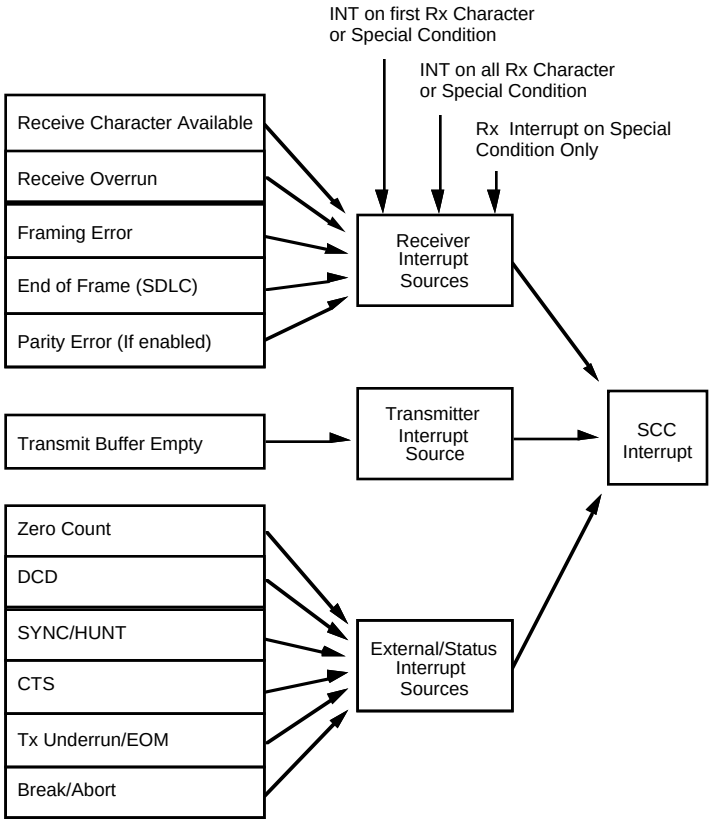



**Figure 2-9. ESCC Interrupt Sources**

***ESCC:***
***The receive interrupt request is either caused by a receive character available or a special condition. When the receive character available interrupt is generated, it is dependent on WR7' bit D3. If WR7' D3=0, the receive character available interrupt is generated when one character is loaded into the FIFO and is ready to be read. If WR7' D3=1, the receive character available interrupt is generated when four bytes are available to be read in the receive data FIFO. The programmed value of WR7' D5 also affects how DMA requests are generated. See Section 2.5 for details.***

**Note:** If the ESCC is used in SDLC mode, it enables the SDLC Status FIFO to affect how receive interrupts are generated. If this feature is used, read Section 4.4.3 on the SDLC Anti-Lock Feature.

The special conditions are Receive FIFO overrun, CRC/framing error, end of frame, and parity. If parity is included as a special condition, it is dependent on WR1 D2. The special condition status can be read from RR1.

On the NMOS/CMOS versions, set the IP bit whenever the transmit buffer becomes empty. This means that the transmit buffer was full before the transmit IP can be set.

***ESCC:***
***The transmit interrupt request has only one source and is dependent on WR7' D5. If the IP bit WR7' D5=0, it is set when the transmit buffer becomes completely empty. If IP bit WR7' D5=1, the transmit interrupt is generated when the entry location of the FIFO is empty. Note that in both cases the transmit interrupt is not set until after the first character is written to the ESCC.***

For more information on Transmit Interrupts, see Section 2.4.8 for details.

The External/status interrupts have several sources which may be individually enabled in WR15. The sources are zero count, /DCD, Sync/Hunt, /CTS, transmitter underrun/EOM and Break/Abort.

## 2.4.4 Interrupt Control

In addition to the MIE bit that enables or disables all SCC interrupts, each source of interrupt in the SCC has three control/status bits associated with it. They are the Interrupt Enable (IE), Interrupt Pending (IP), and Interrupt-Under-Service (IUS). Figure 2-10 shows the SCC interrupt structure.

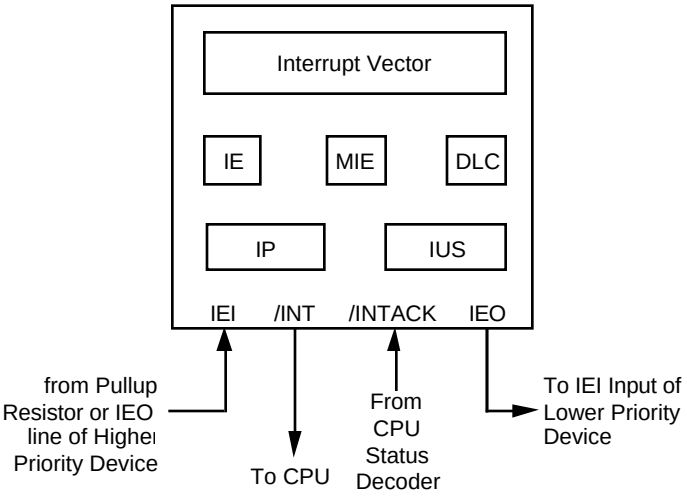



**Figure 2-10. Peripheral Interrupt Structure**

Figure 2-11 shows the internal priority resolution method to allow the highest priority interrupt to be serviced first. Lower priority devices on the external daisy chain can be prevented from requesting interrupts via the Disable Lower Chain bit in WR9 D2.

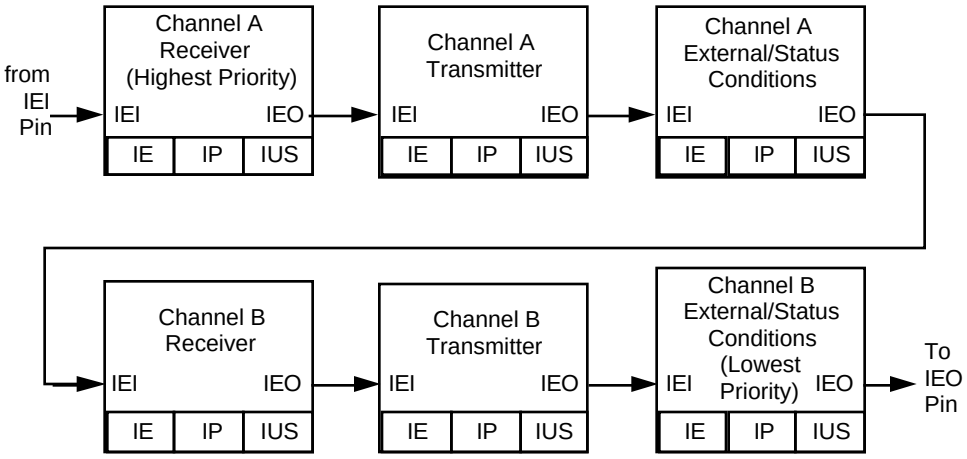



**Figure 2-11. Internal Priority Resolution**

## 2.4 INTERFACE PROGRAMMING (Continued)

### 2.4.4.1 Master Interrupt Enable Bit

The Master Interrupt Enable (MIE) bit, WR9 D3, must be set to enable the SCC to generate interrupts. The MIE bit should be set after initializing the SCC registers and enabling the individual interrupt enables. The SCC requests an interrupt by asserting the /INT pin Low from its open-drain state only upon detection that one of the enabled interrupt conditions has been detected.

### 2.4.4.2 Interrupt Enable Bit

The Interrupt Enable (IE) bits control interrupt requests from each interrupt source on the SCC. If the IE bit is set to 1 for an interrupt source, that source may generate an interrupt request, providing all of the necessary conditions are met. If the IE bit is reset, no interrupt request is generated by that source. The transmit interrupt IE bit is WR1 D1. The receive interrupt IE bits are WR1 D3 and D4. The external status interrupts are individually enabled in WR15 with the master external status interrupt enable in WR1 D0. Reminder: The MIE bit, WR9 D3, must be set for any interrupt to occur.

### 2.4.4.3 Interrupt Pending Bit

The Interrupt Pending (IP) bit for a given source of interrupt is set by the presence of an interrupt condition in the SCC. It is reset directly by the processor, or indirectly by some action that the processor may take. If the corresponding IE bit is not set, the IP for that source of interrupt will never be set. The IP bits in the SCC are read only via RR3 as shown in Figure 2-12.

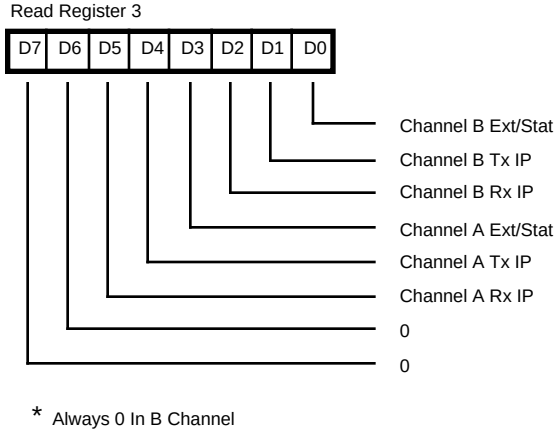



**Figure 2-12. RR3 Interrupt Pending Bits**

### 2.4.4.4 Interrupt-Under-Service Bit

The Interrupt-Under-Service (IUS) bits are completely hidden from the processor. An IUS bit is set during an interrupt acknowledge cycle for the highest priority IP. On the CMOS or ESCC, the IUS bits can be set by either a hardware acknowledge cycle with the /INTACK pin or through software if WR9 D5=1 and then reading RR2.

The IUS bits control the operation of internal and external daisy-chain interrupts. The internal daisy chain links the six sources of interrupt in a fixed order, chaining the IUS bit of each source. If an internal IUS bit is set, all lower priority interrupt requests are masked off; during an interrupt acknowledge cycle the IP bits are also gated into the daisy chain. This ensures that the highest priority IP selected has its IUS bit set. At the end of an interrupt service routine, the processor must issue a Reset Highest IUS command in WR0 to re-enable lower priority interrupts. This is the only way, short of a software or hardware reset, that an IUS bit may be reset.

**Note:** It is not necessary to issue the Reset Highest IUS command in the interrupt service routine, since the IUS bits can only be set by an interrupt acknowledge if no hardware acknowledge or software acknowledge cycle (not with NMOS) is executed. The only exception is when the SDLC Frame Status FIFO (not with NMOS) is enabled and "receive interrupt on special condition only" is used. See section 4.4.3 for more details on this mode.

### 2.4.4.5 Disable Lower Chain Bit

The Disable Lower Chain (DLC) bit in WR9 (D2) is used to disable all peripherals in a lower position on the external daisy chain. If WR9 D2=1, the IEO pin is driven Low and prevents lower priority devices from generating an interrupt request. Note that the IUS bit, when set, will have the same effect, but is not controllable through software.

## 2.4.5 Daisy-Chain Resolution

The six sources of interrupt in the SCC are prioritized in a fixed order via a daisy chain; provision is made, via the IEI and IEO pins, for use of an external daisy chain as well. All Channel A interrupts are higher priority than any Channel B interrupts, with the receiver, transmitter, and External/Status interrupts prioritized in that order within each channel. The SCC requests an interrupt by pulling the /INT pin Low from its open-drain state. This is controlled by the IP bits and the IEI input, among other things. A flowchart of the interrupt sequence for the SCC is shown in Figure 2-13.

The internal daisy chain links the six sources of interrupt in a fixed order, chaining the IUS bits for each source. While an IUS bit is set, all lower priority interrupt requests are masked off, thus preventing lower priority interrupts, but still allowing higher priority interrupts to occur. Also, during an interrupt acknowledge cycle the IP bits are gated into the daisy chain. This insures that the highest priority IP is selected to set IUS. The internal daisy chain may be controlled by the MIE bit in WR9. This bit, when reset, has the same effect as pulling the IEI pin Low, thus disabling all interrupt requests.

### 2.4.5.1 External Daisy-Chain Operations

The SCC generates an interrupt request by pulling /INT Low, but only if such interrupt requests are enabled (IE is 1, MIE is 1) and all of the following conditions occur:

- IP is set without a higher priority IUS being set
- No higher priority IUS is being set
- No higher priority interrupt is being serviced (IEI is High)
- No interrupt acknowledge transaction is taking place

IEO is not pulled Low by the SCC at this time, but instead continues to follow IEI until an interrupt acknowledge transaction occurs. Some time after /INT has been pulled Low, the processor initiates an Interrupt Acknowledge transaction. Between the time the SCC recognizes that an Interrupt Acknowledge cycle is in progress and the time during the acknowledge that the processor requests an interrupt vector, the IEI/IEO daisy chain settles. Any peripheral in the daisy chain having an Interrupt Pending (IP is 1) or an Interrupt-Under-Service (IUS is 1) holds its IEO line Low and all others make IEO follow IEI.

When the processor requests an interrupt vector, only the highest priority interrupt source with a pending interrupt (IP is 1) has its IEI input High, its IE bit set to 1, and its IUS bit set to 0. This is the interrupt source being acknowledged, and at this point it sets its IUS bit to 1. If its NV bit is 0, the SCC identifies itself by placing the interrupt vector from WR2 on the data bus. If the NV bit is 1, the SCC data bus remains floating, allowing external logic to supply a vector. If the VIS bit in the SCC is 1, the vector also contains status information, encoded as shown in Table 2-9, which further describes the nature of the SCC interrupt.

**Table 2-9. Interrupt Vector Modification**

| V3 | V2 | V1 | Status High/Status Low = 0 |
|---|---|---|---|
| **V4** | **V5** | **V6** | **Status High/Status Low = 1** |
| 0 | 0 | 0 | Ch B Transmit Buffer Empty |
| 0 | 0 | 1 | Ch B External/Status Change |
| 0 | 1 | 0 | Ch B Receive Character Avail |
| 0 | 1 | 1 | Ch B Special Receive Condition |
| 1 | 0 | 0 | Ch A Transmit Buffer Empty |
| 1 | 0 | 1 | Ch A External/Status Change |
| 1 | 1 | 0 | Ch A Receive Character Avail |
| 1 | 1 | 1 | Ch A Special Receive Condition |

If the VIS bit is 0, the vector held in WR2 is returned without modification. If the SCC is programmed to include status information in the vector, this status may be encoded and placed in either bits 1-3 or in bits 4-6. This operation is selected by programming the Status High/Status Low bit in WR9. At the end of the interrupt service routine, the processor should issue the Reset Highest IUS command to unlock the daisy chain and allow lower priority interrupt requests. The IP is reset during the interrupt service routine, either directly by command or indirectly through some action taken by the processor. The external daisy chain may be controlled by the DLC bit in WR9. This bit, when set, forces IEO Low, disabling all lower priority devices.

## 2.4 INTERFACE PROGRAMMING (Continued)

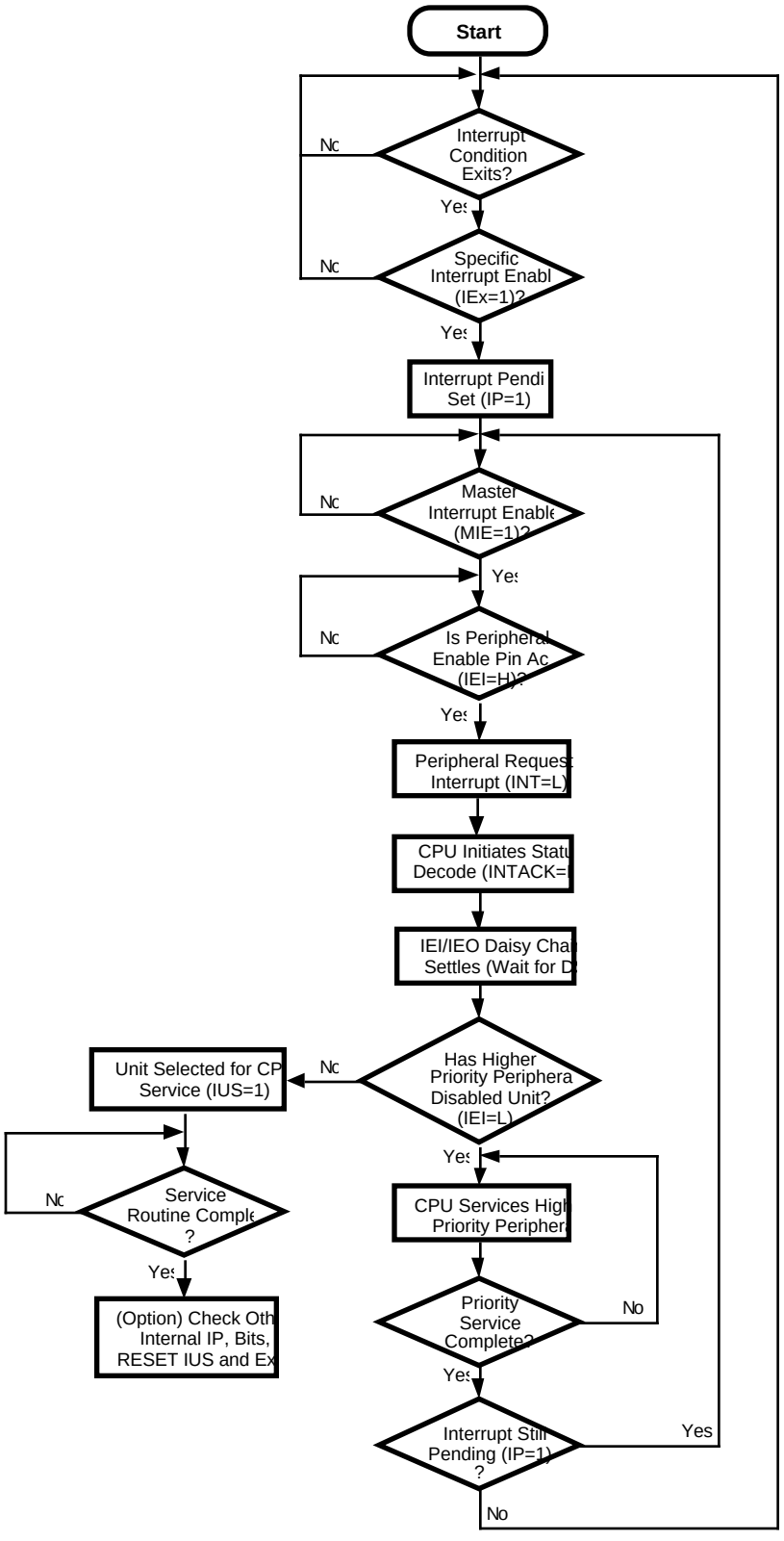



**Figure 2-13. Interrupt Flow Chart (for each interrupt source).**

## 2.4.6 Interrupt Acknowledge

The SCC is flexible with its interrupt method. The interrupt may be acknowledged with a vector transferred, acknowledged without a vector, or not acknowledged at all.

### 2.4.6.1 Interrupt Without Acknowledge

In this mode, the Interrupt Acknowledge signal does not have to be generated. This allows a simpler hardware design that does not have to meet the interrupt acknowledge timing. Soon after the INT goes active, the interrupt controller jumps to the interrupt routine. In the interrupt routine, the code must read RR2 from Channel B to read the vector including status. When the vector is read from Channel B, it always includes the status regardless of the VIS bit (WR9 bit 0). The status given will decode the highest priority interrupt pending at the time it is read. The vector is not latched so that the next read could produce a different vector if another interrupt occurs. The register is disabled from change during the read operation to prevent an error if a higher interrupt occurs exactly during the read operation.

Once the status is read, the interrupt routine must decode the interrupt pending, and clear the condition. Removing the interrupt condition clears the IP and brings /INT inactive (open-drain), as long as there are no other IP bits set. For example, writing a character to the transmit buffer clears the transmit buffer empty IP.

When the interrupt IP, decoded from the status, is cleared, RR2 can be read again. This allows the interrupt routine to clear all of the IP's within one interrupt request to the CPU.

### 2.4.6.2 Interrupt With Acknowledge

After the SCC brings /INT active, the CPU can respond with a hardware acknowledge cycle by bringing /INTACK active. After enough time has elapsed to allow the daisy chain to settle (see AC Spec #38), the SCC sets the IUS bit for the highest priority IP. If the No Vector bit is reset (WR9 D1=0), the SCC then places the interrupt vector on the data bus during a read. To speed the interrupt response time, the SCC can modify 3 bits in the vector to indicate the source of the interrupt. To include the status, the VIS bit, WR9 D0, is set. The service routine must then clear the interrupting condition. For example, writing a character to the transmit buffer clears the transmit buffer empty IP. After the interrupting condition is cleared, the routine can read RR3 to determine if any other IP's are set and take the appropriate action to clear them. At the end of the interrupt routine, a Reset IUS command (WR0) is issued to unlock the daisy chain and allow lower-priority interrupt requests. This is the only way, short of a software or hardware reset, that an IUS bit is reset.

If the No Vector bit is set (WR9 D1=1), the SCC will not place the vector on the data bus. An interrupt controller must then vector the code to the interrupt routine. The interrupt routine reads RR2 from Channel B to read the status. This is similar to an interrupt without an acknowledge, except the IUS is set and the vector will not change until the Reset IUS command in RR0 is issued.

### 2.4.6.3 Software Interrupt Acknowledge (CMOS/ESCC)

An interrupt acknowledge cycle can be done in software for those applications which use an external interrupt controller or which cannot generate the /INTACK signal with the required timing. If WR9 D5 is set, reading register two, RR2, results in an interrupt acknowledge cycle to be executed internally. Like a hardware INTACK cycle,
a software acknowledge causes the /INT pin to return High, the IEO pin to go Low and the IUS latch to be set for the highest priority interrupt pending.

As when the hardware /INTACK signal is used, a software acknowledge cycle requires that a Reset Highest IUS command be issued in the interrupt service routine. If RR2 is read from Channel A, the unmodified vector is returned. If RR2 is read from Channel B, then the vector is modified to indicate the source of the interrupt. The Vector Includes Status (VIS) and No Vector (NV) bits in WR9 are ignored when bit D5 is set to 1.

## 2.4.7 The Receiver Interrupt

The sources of receive interrupts consist of Receive Character Available and Special Receive Condition. The Special Receive Condition can be subdivided into Receive Overrun, Framing Error (Asynchronous) or End of Frame (SDLC). In addition, a parity error can be a special receive condition by programming.

As shown in Figure 2-14, Receive Interrupt mode is controlled by three bits in WR1. Two of these bits, D4 and D3, select the interrupt mode; the third bit, D2, is a modifier for the various modes. On the ESCC, WR7' bit D2 affects the receiver interrupt operation mode as well. If the interrupt capability of the receiver in the SCC is not required, polling may be used. This is selected by disabling receive interrupts and polling the Receiver Character Available bit in RR0. When this bit indicates that a received character has reached the exit location (CPU side) of the FIFO, the status in RR1 should be checked and then the data should be read. If status is checked, it must be done before the data is read, because the act of reading the data pops both the data and error FIFOs. Another way of polling SCC is to enable one of the interrupt modes and then reset the MIE bit in WR9. The processor may then poll the IP bits in RR3A to determine when receive characters are available.

## 2.4 INTERFACE PROGRAMMING (Continued)

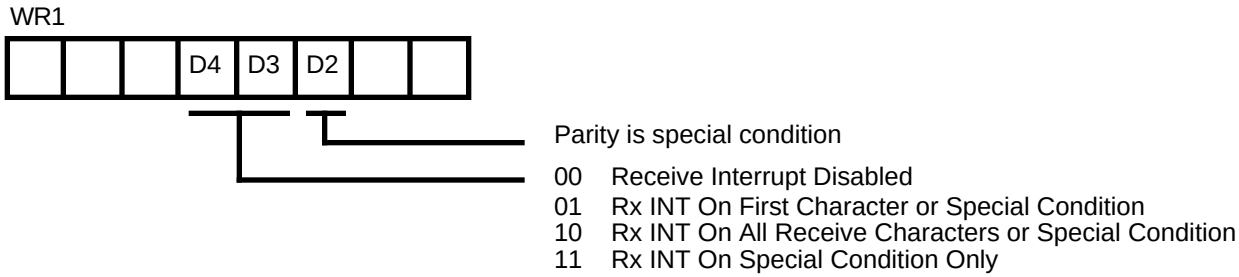



**Figure 2-14. Write Register 1 Receive Interrupt Mode Control**

#### 2.4.7.1 Receive Interrupt on the ESCC

On the ESCC, one other bit, WR7' bit D2, also affects the interrupt operation.

WR7' D3=0, a receive interrupt is generated when one byte is available in the FIFO. This mode is selected after reset and maintains compatibility with the SCC. Systems with a long interrupt response time can use this mode to generate an interrupt when one byte is received, but still allow up to seven more bytes to be received without an overrun error. By polling the Receive Character Available bit, RR0 D0, and reading all available data to empty the FIFO before exiting the interrupt service routine, the frequency of interrupts can be minimized.

WR7' D3=1, the ESCC generates an interrupt when there are four bytes in the Receive FIFO or when a special condition is received. By setting this bit, the ESCC generates a receive interrupt when four bytes are available to read from the FIFO. This allows the CPU not to be interrupted until at least four bytes can be read from the FIFO, thereby minimizing the frequency of receive interrupts. If four or more bytes remain in the FIFO when the Reset Highest IUS command is issued at the end of the service routine, another receive interrupt is generated.

When a special receive condition is detected in the top four bytes, a special receive condition interrupt is generated immediately. This feature is intended to be used with the Interrupt On All Receive Characters and Special Condition mode. This is especially useful in SDLC mode because the characters are contiguous and the reception of the closing flag immediately generates a special receive interrupt. The generation of receive interrupts is described in the following two cases:

**Case 1:** Four Bytes Received with No Errors. A receive character available interrupt is triggered when the four bytes in receive data FIFO (from the exit side) are full and no special conditions have been detected. Therefore, the interrupt service routine can read four bytes from the data FIFO without having to read RR1 to check for error conditions.

**Case 2:** Data Received with Error Conditions. When any of the four bytes from the exit side in the receive error FIFO indicate an error has been detected, a Special Receive condition interrupt is triggered without waiting for the byte to reach the top of the FIFO. In this case, the interrupt service routine must read RR1 first before reading each data byte to determine which byte has the special receive condition and then take the appropriate action. Since, in this mode, the status must be checked before the data is read, the data FIFO is not locked and the Error Reset command is not necessary.

**Note:** The above cases assume that the receive IUS bit is reset to zero in order for an interrupt to be generated.

WR7' D3 should be written zero when using Interrupt on First Character and Special Condition or Interrupt on Special Condition Only. See the description for Interrupt on All Characters or Special Condition mode for more details on this feature.

**Note:** The Receive Character Available Status bit, RR0 D0, indicates if at least one byte is available in the Receive FIFO, independent of WR7' D3. Therefore, this bit can be polled at any time for status if there is data in the Receive FIFO.

#### 2.4.7.2 Receive Interrupts Disabled

This mode prevents the receiver from requesting an interrupt. It is used in a polled environment where either the status bits in RR0 or the modified vector in RR2 (Channel B) is read. Although the receiver interrupts are disabled, the interrupt logic can still be used to provide status.

When these bits indicate that a received character has reached the exit location of the FIFO, the status in RR1 should be checked and then the data should be read. If status is to be checked, it must be done before the data is read, because the act of reading the data pops both the data and error FIFOs.

#### 2.4.7.3 Receive Interrupt on First Character or Special Condition

This mode is designed for use with DMA transfers of the receive characters. The processor is interrupted when the SCC receives the first character of a block of data. It reads the character and then turns control over to a DMA device to transfer the remaining characters. After this mode is selected, the first character received, or the first character already stored in the FIFO, sets the receiver IP. This IP is reset when this character is removed from the SCC.

No further receive interrupts occur until the processor issues an Enable Interrupt on Next Receive Character command in WR0 or until a special receive condition occurs. The correct sequence of events when using this mode is to first select the mode and wait for the receive character available interrupt. When the interrupt occurs, the processor should read the character and then enable the DMA to transfer the remaining characters.

***ESCC:***
***WR7' bit D3 should be reset to zero in this mode.***

A special receive condition interrupt may occur any time after the first character is received, but is guaranteed to occur after the character having the special condition has been read. The status is not lost in this case, however, because the FIFO is locked by the special condition. In the interrupt service routine, the processor should read RR1 to obtain the status, and may read the data again if necessary. The FIFO is unlocked by issuing an Error Reset command in WR0. If the special condition was End-of-Frame, the processor should now issue the Enable Interrupt on Next Receive Character command to prepare for the next frame. The first character interrupt and special condition interrupt are distinguished by the status included in the interrupt vector. In all other respects they are identical, including sharing the IP and IUS bits.

#### 2.4.7.4 Interrupt on All Receive Characters or Special Condition

This mode is designed for an interrupt driven system. In this mode, the NMOS/CMOS version and the ESCC with WR7' D3=0 sets the receive IP when a received character is shifted into the exit location of the FIFO. This occurs whether or not it has a special receive condition. This includes characters already in the FIFO when this mode is selected. In this mode of operation the IP is reset when the character is removed from the FIFO, so if the processor requires status for any characters, this status must be read before the data is removed from the FIFO.

On the ESCC with D3=1, four bytes are accumulated in the Receive FIFO before an interrupt is generated (IP is set), and reset when the number of the characters in the FIFO is less than four.

The special receive conditions are identical to those previously mentioned, and as before, the only difference between a "receive character available" interrupt and a "special receive condition" interrupt is the status encoded in the vector. In this mode a special receive condition does not lock the receive data FIFO so that the service routine must read the status in RR1 before reading the data.

At moderate to high data rates where the interrupt overhead is significant, time can usually be saved by checking for another character before exiting the service routine. This technique eliminates the interrupt acknowledge and the status processing, saving time, but care must be exercised because this receive character must be checked for special receive conditions before it is removed from the SCC.

#### 2.4.7.5 Receive Interrupt on Special Conditions

This mode is designed for use when a DMA transfers all receive characters between memory and the SCC. In this mode, only receive characters with special conditions will cause the receive IP to be set. All other characters are assumed to be transferred via DMA. No special initialization sequence is needed in this mode. Usually, the DMA is initialized and enabled, then this mode is selected in the SCC. A special receive condition interrupt may occur at any time after this mode is selected, but the logic guarantees that the interrupt will not occur until after the character with the special condition has been read from the SCC. The special condition locks the FIFO so that the status is valid when read in the interrupt service routine, and it guarantees that the DMA will not transfer any characters until the special condition has been serviced.

In the service routine, the processor should read RR1 to obtain the status and unlock the FIFO by issuing an Error Reset command. DMA transfer of the receive characters then resumes. Figure 2-15 shows the special conditions interrupt service routine.

**Note:** On the CMOS and ESCC, if the SDLC Frame Status FIFO is being used, please refer to Section 4.4.3 on the FIFO anti-lock feature.

**Note:** Special Receive Condition interrupts are generated *after* the character is read from the FIFO, *not* when the special condition is first detected. This is done so that when using receive interrupt on first or Special Condition or Special Condition Only, data is directly read out of the data FIFO without checking the status first. If a special condition interrupted the CPU when first detected, it would be necessary to read RR1 before each byte in the FIFO to determine which byte had the special condition. Therefore,

## 2.4 INTERFACE PROGRAMMING (Continued)

by not generating the interrupt until after the byte has been read and then locking the FIFO, only one status read is necessary. A DMA can be used to do all data transfers (otherwise, it would be necessary to disable the DMA to allow the CPU to read the status on each byte). Consequently, since the special condition locks the FIFO to preserve the status, it is necessary to issue the Error Reset command to unlock it. Only the exit location of the FIFO is locked allowing more data to be received into the other bytes of the Receive FIFO.

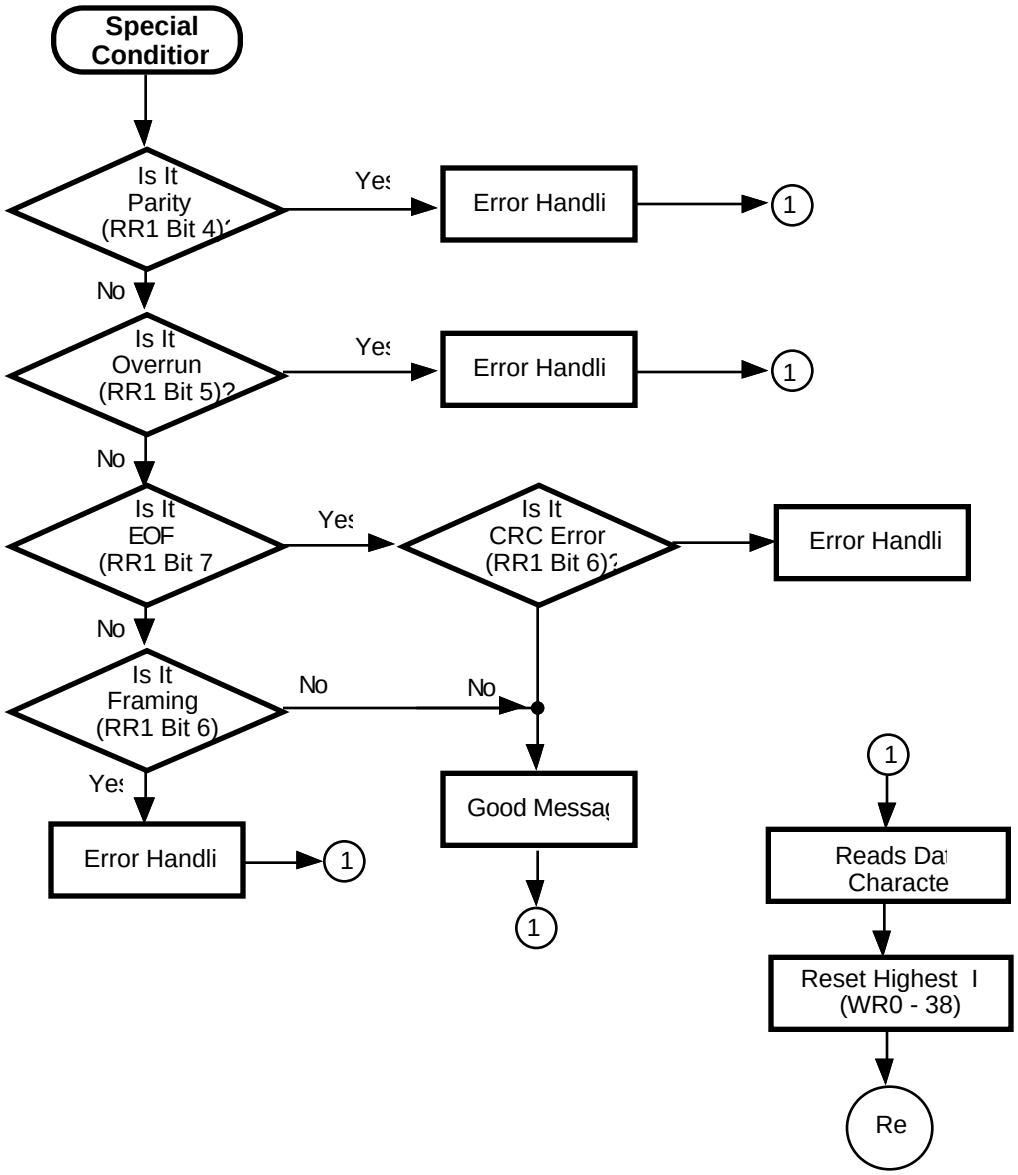



**Figure 2-15. Special Conditions Interrupt Service Flow**

## 2.4.8 Transmit Interrupts and Transmit Buffer Empty Bit

Transmit interrupts are controlled by Transmit Interrupt Enable bit (D1) in WR1. If the interrupt capabilities of the SCC are not required, polling may be used. This is selected by disabling transmit interrupts and polling the Transmit Buffer Empty bit (TBE) in RR0. When the TBE bit is set, a character may be written to the SCC without fear of writing over previous data. Another way of polling the SCC is to enable transmit interrupts and then reset Master Interrupt Enable bit (MIE) in WR9. The processor may then poll the IP bits in RR3A to determine when the transmit buffer is empty. Transmit interrupts should also be disabled in the case of DMA transfer of the transmitted data.

Because the depth of the transmitter buffer is different between the NMOS/CMOS version of the SCC and ESCC, generation of the transmit interrupt is slightly different. The following subsections describe transmit interrupts.

**Note:** For all interrupt sources, the Master Interrupt Enable (MIE) bit, WR9 bit D3, must be set for the device to generate a transmit interrupt.

### 2.4.8.1 Transmit Interrupts and Transmit Buffer Empty Bit on the NMOS/CMOS

The NMOS/CMOS version of the SCC only has a one byte deep transmit buffer. The status of the transmit buffer can be determined through TBE bit in RR0, bit D2, which shows whether the transmit buffer is empty or not. After a hardware reset (including a hardware reset by software), or a channel reset, this bit is set to 1.

While transmit interrupts are enabled, the NMOS/CMOS version sets the Transmit Interrupt Pending (TxIP) bit whenever the transmit buffer becomes empty. This means that the transmit buffer must be full before the TxIP can be set. Thus, when transmit interrupts are first enabled, the TxIP will not be set until after the first character is written to the NMOS/CMOS. In synchronous modes, one other condition can cause the TxIP to be set. This occurs at the end of a transmission after the CRC is sent. When the last bit of the CRC has cleared the Transmit Shift Register and the flag or sync character is loaded into the Transmit Shift Register, the NMOS/CMOS version sets the TxIP and TBE bit. Data for a second frame or block transmission may be written at this time.

The TxIP is reset either by writing data to the transmit buffer or by issuing the Reset Tx Int command in WR0. Ordinarily, the response to a transmit interrupt is to write more data to the device; however, the Reset Tx Int command should be issued in lieu of data at the end of a frame or a block of data where the CRC is to be sent next.

**Note:** A transmit interrupt may indicate that the packet has terminated illegally, with the CRC byte(s) overwritten by the data. If the transmit interrupt occurs after the first CRC byte is loaded into the Transmit Shift Register, but before the last bit of the second CRC byte has cleared the Transmit Shift Register, then data was written while the CRC was being sent.

### 2.4.8.2 Transmit Interrupt and Transmit Buffer Empty bit on the ESCC

The ESCC has a 4-byte deep Transmit FIFO, while the NMOS/CMOS SCC is just 1-byte deep. For this reason, the generation of transmit interrupts is slightly different from that of the NMOS/CMOS SCC version. The ESCC has two modes of transmit interrupt generation, which are programmed by bit D5 of WR7'. One transmit mode generates interrupts when the entry location (the location the CPU writes data) of the Transmit FIFO is empty. This allows the ESCC response to be tailored to system requirements for the frequency of interrupts and the interrupt response time. On the other hand, the Transmit Buffer Empty (TBE) bit on the ESCC will respond the same way in each mode, in which the bit will become set when the entry location of the Transmit FIFO is empty. The TBE bit is not directly related to the transmit interrupt status nor the state of WR7' bit D5.

When WR7' D5=1 (the default case), the ESCC will generate a transmit interrupt when the Transmit FIFO becomes completely empty. The transmit interrupt occurs when the data in the exit location of the Transmit FIFO loads into the Transmit Shift Register and the Transmit FIFO becomes completely empty. This mode minimizes the frequency of transmit interrupts by writing 4 bytes to the Transmit FIFO upon each entry to the interrupt will become set when WR7' D5=1. The TBE bit RR0 bit D2 will become set whenever the entry location of the Transmit FIFO becomes empty. The TBE bit will reset when the entry location becomes full. The TBE bit in a sense translates to meaning "Transmit Buffer Not Full" for the ESCC only, as the TBE bit will become set whenever the entry location of the Transmit FIFO becomes empty. This bit may be polled at any time to determine if a byte can be written to the FIFO. Figure 2-17 illustrates when the TBE bit will become set. WR7' bit D5 is set to one by a hardware or channel reset.

When WR7' D5=0, the TxIP bit is set when the entry location of the Transmit FIFO becomes empty. In this mode, only one byte is written to the Transmit FIFO at a time for each transmit interrupt. The ESCC will generate transmit interrupts when there are 3 or fewer bytes in the FIFO, and will continue to do so until the FIFO is filled. When WR7' D5=0, the transmit interrupt is reset momentarily when data is loaded into the entry location of the Transmit FIFO. Transmit interrupt is not generated when the entry location of the Transmit FIFO is filled. The transmit interrupt is generated when the data is pushed down the FIFO and the entry location becomes empty (approximately one PCLK time). Figure 2-18 illustrates when the transmit interrupts will become set when WR7' D5=0. Again, the TBE bit is not dependent on the state of WR7'

## 2.4 INTERFACE PROGRAMMING (Continued)

bit D5 nor the transmit interrupt status, and will respond exactly the same way as mentioned above. Figure 2-17 illustrates when the TBE bit will become set.

**Note:** When WR7' D5=0. only one byte is written to the FIFO at a time, when there are three or fewer bytes in FIFO. Thus, for the ESCC multiple interrupts are generated to fill the FIFO. To avoid multiple interrupts, one can poll the TBE bit (RR0 D2) after writing each byte.

While transmit interrupts are enabled, the ESCC sets the TxIP when the transmit buffer reaches the condition programmed in WR7' bit D5. This means that the transmit buffer must have been written to before the TxIP is set. Thus, when transmit interrupts are first enabled, the transmit IP is not set until the programmed interrupting condition is met.

The TxIP is reset either by writing data to the transmit buffer or by issuing the Reset Tx Int Pending command in WR0. Ordinarily, the response to a transmit interrupt is to write more data to the ESCC; however, if there is no more data to be transmitted at that time, it is the end of the frame. The Reset Tx Int command is used to reset the TxIP and clear the interrupt. For example, at the end of a frame or block of data where the CRC is to be sent next, the Reset Tx Int Pending command should be issued after the last byte of data has been written to the ESCC.

In synchronous modes, one other condition can cause the TxIP to be set. This occurs at the end of a transmission after the CRC is sent. When the last bit of the CRC has cleared the Transmit Shift Register and the flag or sync character is loaded into the Transmit Shift Register, the ESCC sets the TxIP. Data for the new frame or block to be transmitted may be written at this time. In this particular case, the Transmit Buffer Empty bit in RR0 and the TxIP are set.

An enhancement to the ESCC from the NMOS/CMOS version is that the CRC has priority over the data, where on the NMOS/CMOS version data has priority over the CRS. This means that on the ESCC the CRC bytes are guaranteed to be sent, even if the data for the next packet has written before the second transmit interrupt, but after the EOM/Underrun condition exists. This helps to increase the system throughput because there is not waiting for the second transmit interrupt. On the NMOS/CMOS version, if the data is written while the CRC is sent, CRC byte(s) are replaced with the flag/sync pattern followed by the data.

Another enhancement of the ESCC is that it latches the transmit interrupt because the CRC is loaded into the Transmit Shift Register even if the transmit interrupt, due to the last data byte, is not yet reset. Therefore, the end of a synchronous frame is guaranteed to generate two transmit interrupts even if a Reset Tx Int Pending command for the data created interrupt is issued after (Time "A" in Figure 2-16) the CRC interrupt had occurred. In this case, two reset Tx Int Pending commands are required. The TxIP is latched if the EOM latch has been reset before the end of the frame.

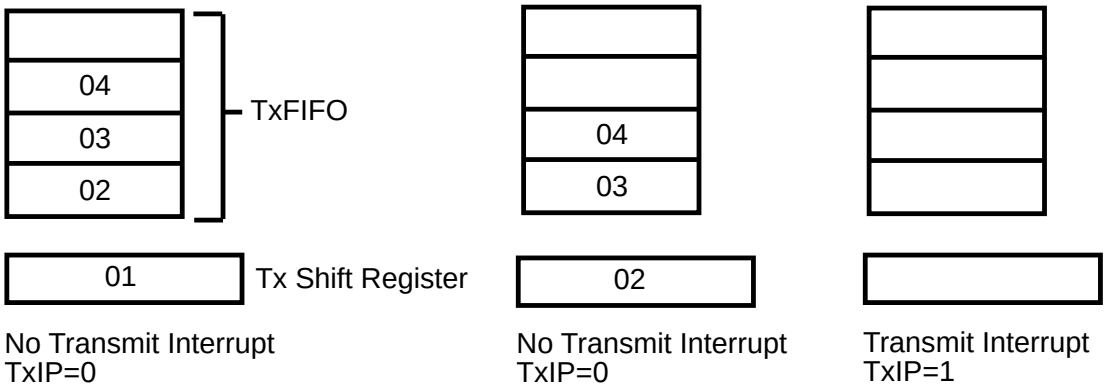



**Figure 2-16. Transmit Interrupt Status When WR7' D5=1 For ESCC**

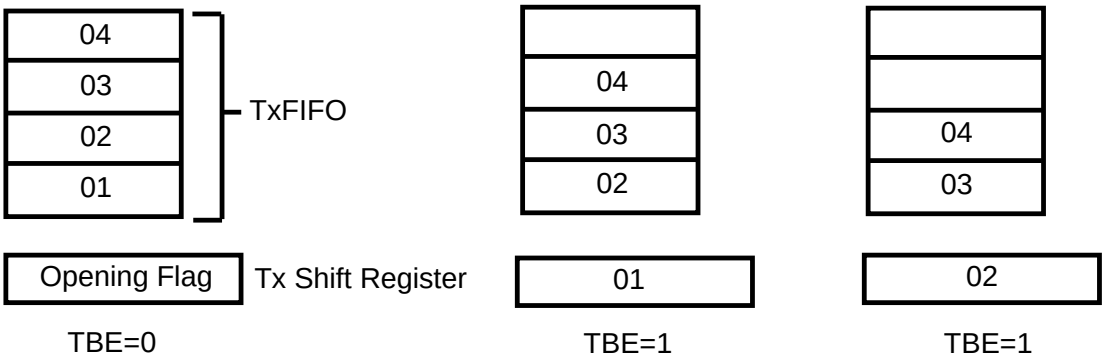



**Figure 2-17. Transmit Buffer Empty Bit Status For ESCC For Both WR7' and WR7' D5=0**

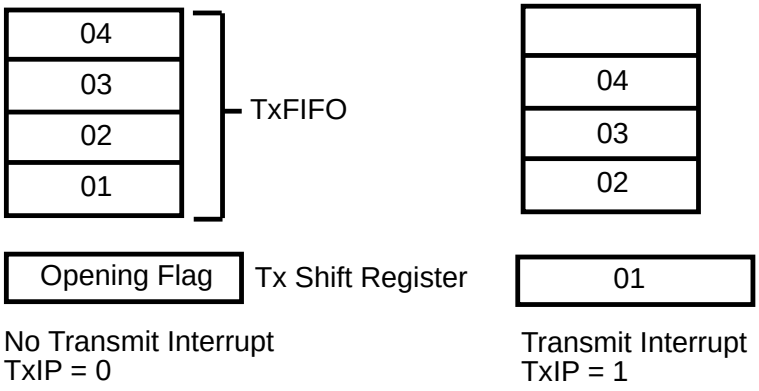



**Figure 2-18. Transmit Interrupt Status When WR7' D5=0 For ESCC**

.

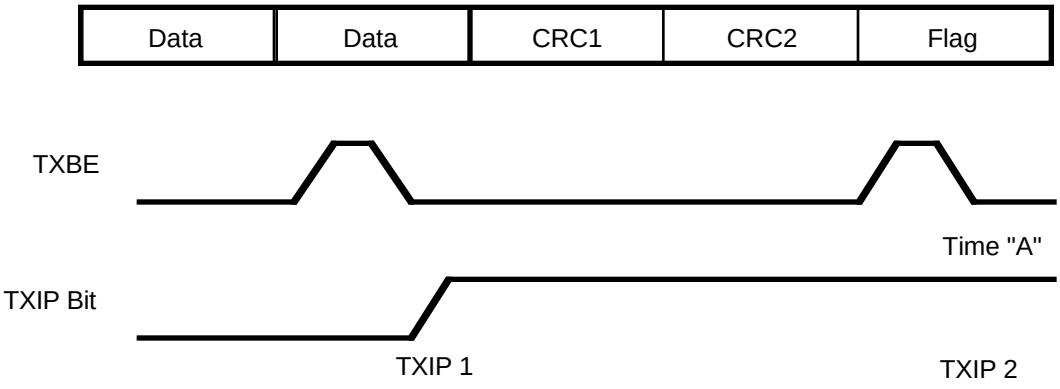



**Figure 2-19. TxIP Latching on the ESCC**

## 2.4 INTERFACE PROGRAMMING (Continued)

### 2.4.8.3 Transmit Interrupt and Tx Underrun/EOM bit in synchronous modes

As described in the section above, the behavior of the NMOS/CMOS version and the ESCC is slightly different, particularly at the end of packet sending. On the NMOS/CMOS version, the data has higher priority over CRC data; writing data before this interrupt would terminate the packet illegally. In this case, the CRC byte(s) are replaced with a Flag or Sync pattern, followed by the data written. On the ESCC, the CRC has priority over the data. That means after the reception of the Underrun/EOM (End Of Message) interrupt, it accepts the data for the next packet without collapsing the packet. On the ESCC, if data was written during the time period described above, the TBE bit (bit D2 of RR0) will *not* be set even if the second TxIP is guaranteed to set when the flag/sync pattern was loaded into the Transmit Shift Register, as mentioned above (Figures 2-17 and 18). Hence, on the ESCC, there is no need to wait for the second TxIP bit to set before writing data for the next packet and reducing the overhead.

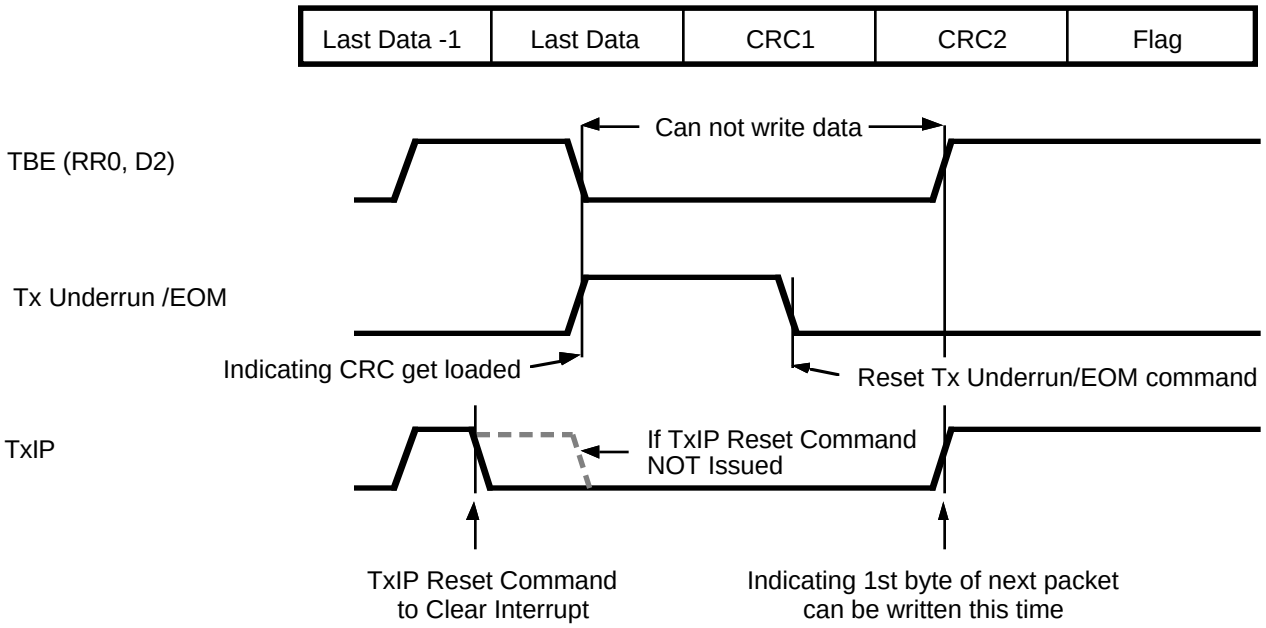



**Figure 2-20. Operation of TBE, Tx Underrun/EOM and TxIP on NMOS/CMOS.**

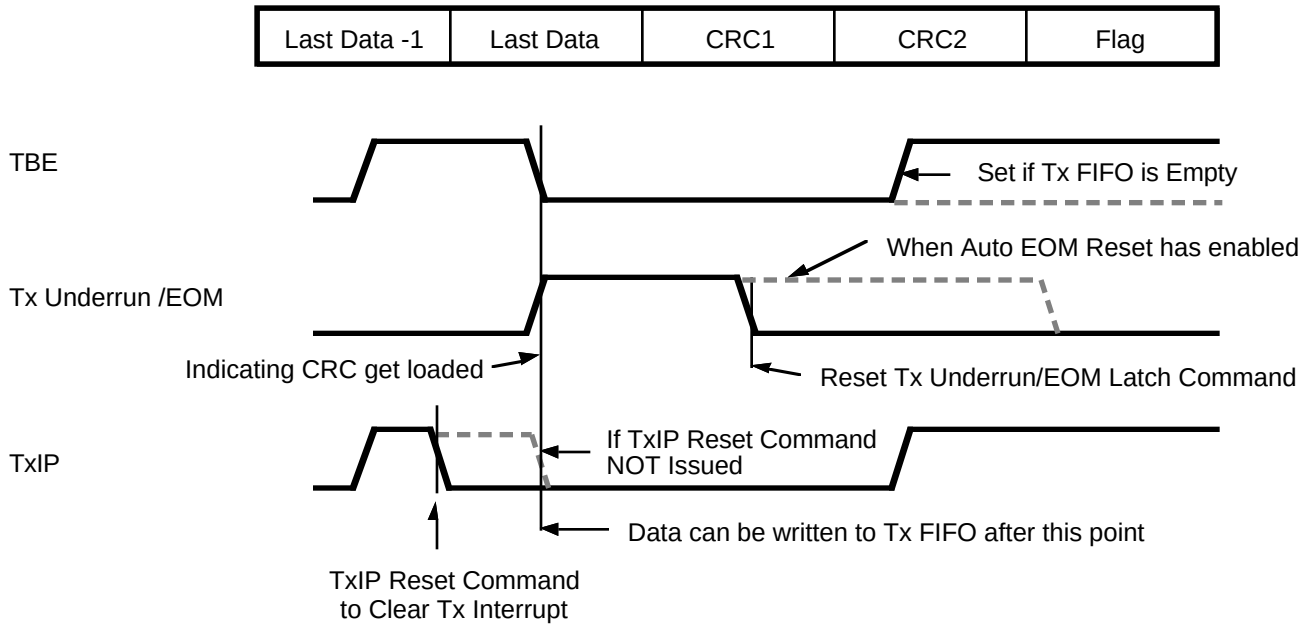



**Figure 2-21. Operation of TBE, Tx Underrun/EOM and TxIP on ESCC**

An example flowchart for processing an end of packet is shown in Figure 2-22. The chart includes the differences in processing between the ESCC and NMOS/CMOS version. In this chart, Tx IP and Underrun/EOM INT can be processed by interrupts or by polling the registers. Note that this flowchart does not have the procedures for interrupt handling, such as saving/restoring of registers to be used in the ISR (Interrupt Service Routine), Reset IUS command, or return from interrupt sequence.

## 2.4 INTERFACE PROGRAMMING (Continued)

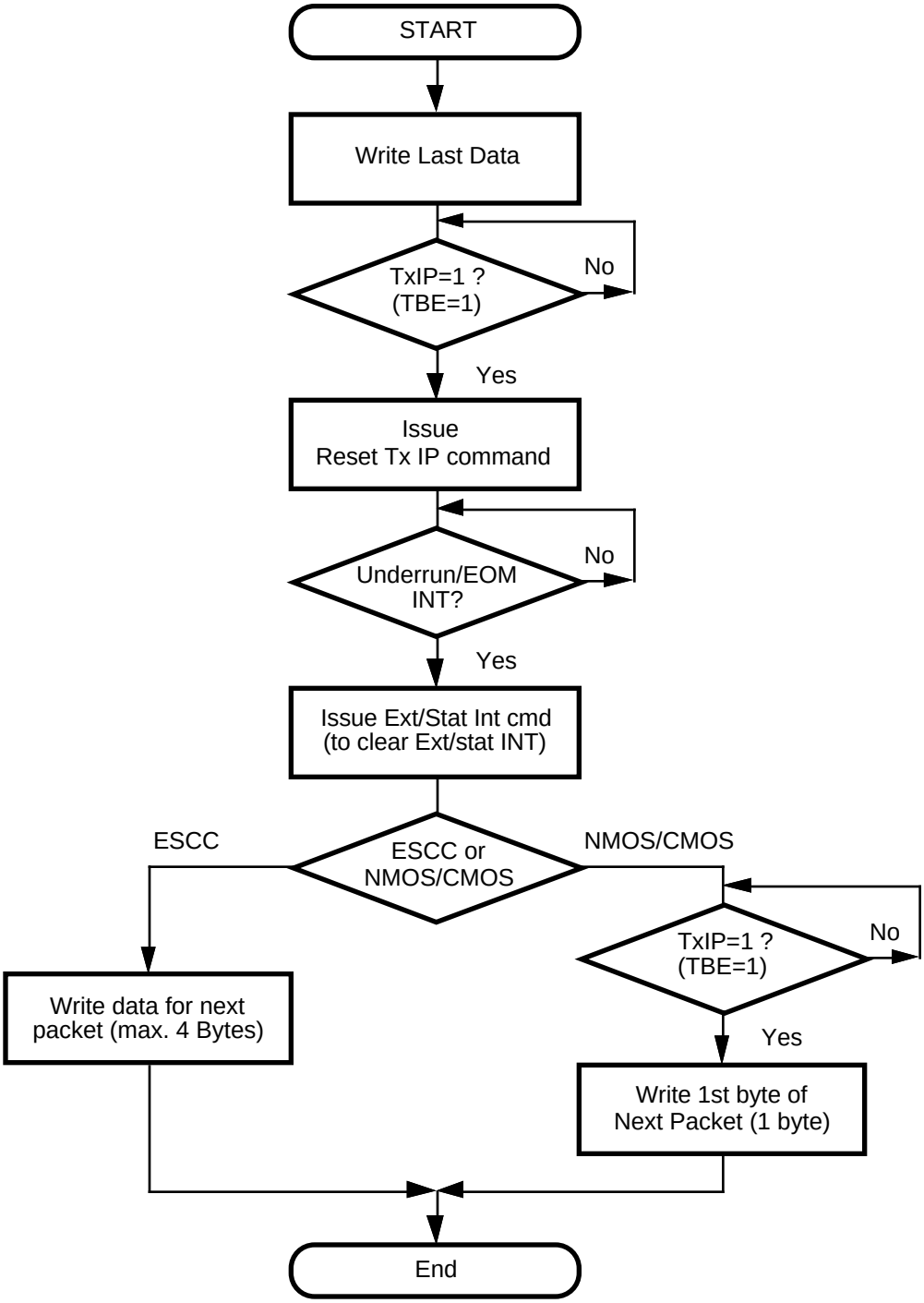



**Figure 2-22. Flowchart example of processing an end of packet**

### 2.4.9 External/Status Interrupts

Each channel has six external/status interrupt conditions: BRG Zero Count, Data Carrier Detect, Sync/Hunt, Clear to Send, Tx Underrun/EOM, and Break/Abort. The master enable for external/status interrupts is D0 of WR1, and the individual enable bits are in WR15. Individual enable bits control whether or not a latch is present in the path from the source of the interrupt to the corresponding status bit in RR0. If the individual enable is set to 0, then RR0 reflects the current unlatched status, and if the individual enable is set to 1, then RR0 reflects the latched status.

The latches for the external/status interrupts are not independent. Rather, they all close at the same time as a result of a state change in one of the sources of enabled external/status interrupts. This is shown schematically in Figure 2-23.

The External/Status IP is set by the closing of the latches and remains set as long as they are closed. In order to determine which condition(s) require service when an external/status interrupt is received, the processor should keep an image of RR0 in memory and update this image each time it executes the external/status service routine.

Thus, a read of RR0 returns the current status for any bits whose individual enable is 0, and either the current state or the latched state of the remainder of the bits. To guarantee the current status, the processor should issue a Reset External/Status interrupts command in WR0 to open the latches. The External/Status IP is set by the closing of the latches and remains set as long as they are closed. If the master enable for the External/Status interrupts is not set, the IP is never set, even though the latches may be present in the signal paths and working as described.

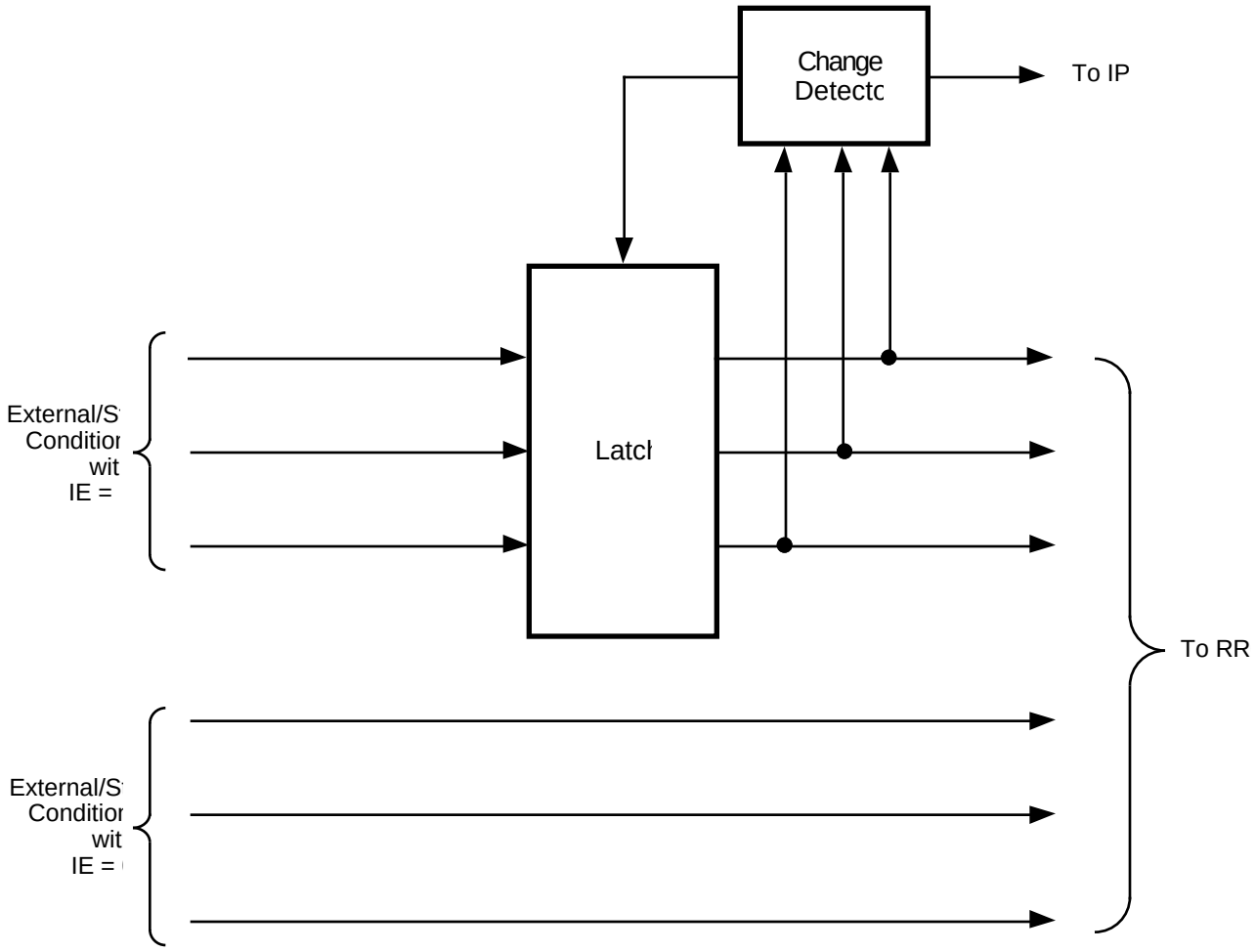



**Figure 2-23. RR0 External/Status Interrupt Operation**

## 2.4 INTERFACE PROGRAMMING (Continued)

Because the latches close on the current status, but give no indication of change, the processor must maintain a copy of RR0 in memory. When the SCC generates an External/Status Interrupt, the processor should read RR0 and determine which condition changed state and take appropriate action. The copy of RR0 in memory is then updated and the Reset External/Status Interrupt command issued. Care must be taken in writing the interrupt service routine for the External/Status interrupts because it is possible for more than one status condition to change state at the same time. All of the latch bits in RR0 should be compared to the copy of RR0 in memory. If none have changed and the ZC interrupt is enabled, the Zero Count condition caused the interrupt.

On the ESCC, the contents of RR0 are latched while reading this register. The ESCC prevents the contents of RR0 from changing while the read cycle is active. On the NMOS/CMOS version, it is possible for the status of RR0 to change while a read is in progress, so it is necessary to read RR0 twice to detect changes that otherwise may be missed. The contents of RR0 are latched on the falling edge of /RD and are updated after the rising edge of /RD.

The operation of the individual enable bits in WR15 for each of the six sources of External/Status interrupts is identical, but subtle differences exist in the operation of each source of interrupt. The six sources are Break/Abort, Underrun/EOM, CTS, DCD, Sync/Hunt and Zero Count. The Break/Abort, Underrun/EOM, and Zero Count conditions are internal to the SCC, while Sync/Hunt may be internal or external, and CTS and DCD are purely external signals. In the following discussions, each source is assumed to be enabled so that the latches are present and the External/Status interrupts are enabled as a whole. Recall that the External/Status IP is set while the latches are closed and that the state of the signal is reflected immediately in RR0 if the latches are not present.

#### 2.4.9.1 Break/Abort

The Break/Abort status is used in asynchronous and SDLC modes, but is always 0 in synchronous modes other than SDLC. In asynchronous modes, this bit is set when a break sequence (null character plus framing error) is detected in the receive data stream, and remains set as long as 0s continue to be received. This bit is reset when a 1 is received. A single null character is left in the Receive FIFO each time that the break condition is terminated. This character should be read and discarded.

In SDLC mode, this bit is set by the detection of an abort sequence which is seven or more contiguous 1s in the receive data stream. The bit is reset when a 0 is received. A received abort forces the receiver into Hunt, which is also an external/status condition. Though these two bits change state at roughly the same time, one or two External/Status Interrupts may be generated as a result. The Break/Abort bit is unique in that both transitions are guaranteed to cause the latches to close, even if another External/Status interrupt is pending at the time these transitions occur. This guarantees that a break or abort will be caught. This bit is undetermined after reset.

#### 2.4.9.2 Transmit Underrun/EOM

The Transmit Underrun/EOM bit is used in synchronous modes to control the transmission of the CRC. This bit is reset by issuing the Reset Transmit Underrun/EOM command in WR0. However, this transition does not cause the latches to close; this occurs only when the bit is set. To inform the processor of this fact, the SCC sets this bit when the CRC is loaded into the Transmit Shift Register. This bit is also set if the processor issues the Send Abort command in WR0. This bit is always set in Asynchronous mode.

***ESCC:***
***The ESCC has been modified so that in SDLC mode this interrupt indicates when more data can be written to the Transmit FIFO. When this interrupt is used in this way, the Automatic SDLC Flag Transmission feature must be enabled (WR7' D0=1). On the ESCC, the Transmit Underrun/EOM interrupt can be used to signal when data for a subsequent frame can be written to the Transmit FIFO which more easily supports the transmission of back to back frames.***

#### 2.4.9.3 CTS/DCD

The CTS bit reports the state of the /CTS input, and the DCD bit reports the status of the /DCD input. Both bits latch on either input transition. In both cases, after the Reset External/Status Interrupt command is issued, if the latches are closed, they remain closed if there is any odd number of transitions on an input; they open if there is an even number of transitions on the input.

#### 2.4.9.4 Zero Count

The Zero Count bit is set when the counter in the baud rate generator reaches a count of 0 and is reset when the counter is reloaded. The latches are closed only when this bit is set to 1. The status in RR0 always reflects the current status. While the Zero count IE bit in WR15 is reset, this bit is forced to 0.

#### 2.4.9.5 Sync/Hunt

There are a variety of ways in which the Sync/Hunt may be set and reset, depending on the SCC's mode of operation. In the Asynchronous mode this bit reports the state of the /SYNC pin, latching on both input transitions. The same is true of External Sync mode. However, if the crystal oscillator is enabled while in Asynchronous mode, this bit will be forced to 0 and the latches will not be closed. Selecting the

crystal option in External Sync mode is illegal, but the result will be the same.

In Synchronous modes other than SDLC, the Sync/Hunt reports the Hunt state of the receiver. Hunt mode is entered when the processor issues the Enter Hunt command in WR3. This forces the receiver to search for a sync character match in the receive data stream. Because both transitions of the Hunt bit close the latches, issuing this command will cause an External/Status interrupt. The SCC resets this bit when character synchronization has been achieved, causing the latches to again be closed.

In these synchronous modes, the SCC will not re-enter the Hunt mode automatically; only the Enter Hunt command will set this bit. In SDLC mode this bit is also set by the Enter Hunt command, but the receiver automatically enters the Hunt mode if an Abort sequence is received. The receiver leaves Hunt upon receipt of a flag sequence. Both transitions of the Hunt bit will cause the latches to be closed. In SDLC mode, the receiver automatically synchronizes on Flag characters. The receiver is in Hunt mode when it is enabled, so the Enter Hunt command is never needed.

#### 2.4.9.6 External/Status Interrupt Handling

If careful attention is paid to details, the interrupt service routine for External/Status interrupts is straightforward. To determine which bit or bits changed state, the routine should first read RR0 and compare it to a copy from memory. For each changed bit, the appropriate action should be taken and the copy in memory updated. The service routine should close with two Reset External/Status interrupt commands to reopen the latches. The copy of RR0 in memory should always have the Zero Count bit set to 0, since this is the state of the bit after the Reset External/Status interrupts command at the end of the service routine. When the processor issues the Reset Transmit Underrun/EOM latch command in WR0, the Transmit Underrun/EOM bit in the copy of RR0 in memory should be reset because this transition does not cause an interrupt.

## 2.5 BLOCK/DMA TRANSFER

The SCC provides a Block Transfer mode to accommodate CPU block transfer functions and DMA controllers. The Block Transfer mode uses the /W//REQ output in conjunction with the Wait/Request bits in Write Register 1. The /W//REQ output can be defined by software as a /WAIT line in the CPU Block Transfer mode or as a /REQ line in the DMA Block Transfer mode. The /DTR//REQ pin can also be programmed through WR14 bit D2 to function as a DMA request for the transmitter.

To a DMA controller, the SCC's /REQ outputs indicate that the SCC is ready to transfer data to or from memory. To the CPU, the /WAIT output indicates that the SCC is not ready to transfer data, thereby requesting the CPU to extend the I/O cycle.

### 2.5.1 Block Transfers

The SCC offers several alternatives for the block transfer of data. The various options are selected by WR1 (bits D7 through D5) and WR14 (bit D2). Each channel in the SCC has two pins which are used to control the block transfer of data. Both pins in each channel may be programmed to act as DMA Request signals. The /W//REQ pin in each channel may be programmed to act as a Wait signal for the CPU. In either mode, it is advisable to select and enable the mode in two separate accesses of the appropriate register. The first access should select the mode and the second access should enable the function. This procedure prevents glitches on the output pins. Reset forces Wait mode, with /W//REQ open-drain.

#### 2.5.1.1 Wait On Transmit

The Wait On Transmit function is selected by setting both D6 and D5 to 0 and then enabling the function by setting D7 of WR1 to 1. In this mode the /W//REQ pin carries the /WAIT signal, and is open-drain when inactive and Low when active. When the processor attempts to write to the transmit buffer when it is full, the SCC asserts /WAIT until the byte is written (Figure 2-24).

## 2.5 BLOCK/DMA TRANSFER (Continued)

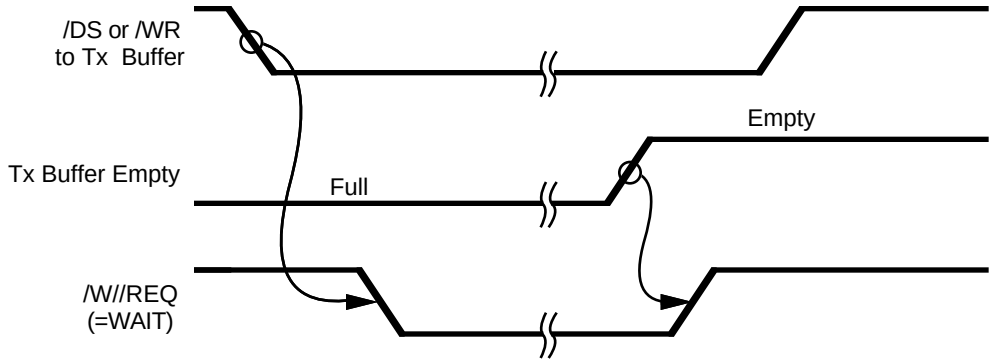



**Figure 2-24. Wait On Transmit Timing**

This allows the use of a block move instruction to transfer the transmit data. In the case of the Z80X30, /WAIT will go active in response to /DS going active, but only if WR8 is being accessed and a write is attempted. In all other cases, /WAIT remains open-drain. In the case of the Z85X30, /WAIT goes active in response to /WR going active, but only if the data buffer is being accessed, either directly or via the pointers. The /WAIT pin is released in response to the falling edge of PCLK. Details of the timing are shown in Figure 2-25.

Care must be taken when using this function, particularly at slow transmission speed. The /WAIT pin stays active as long as the transmit buffer stays full, so there is a possibility that the CPU may be kept waiting for a long period.

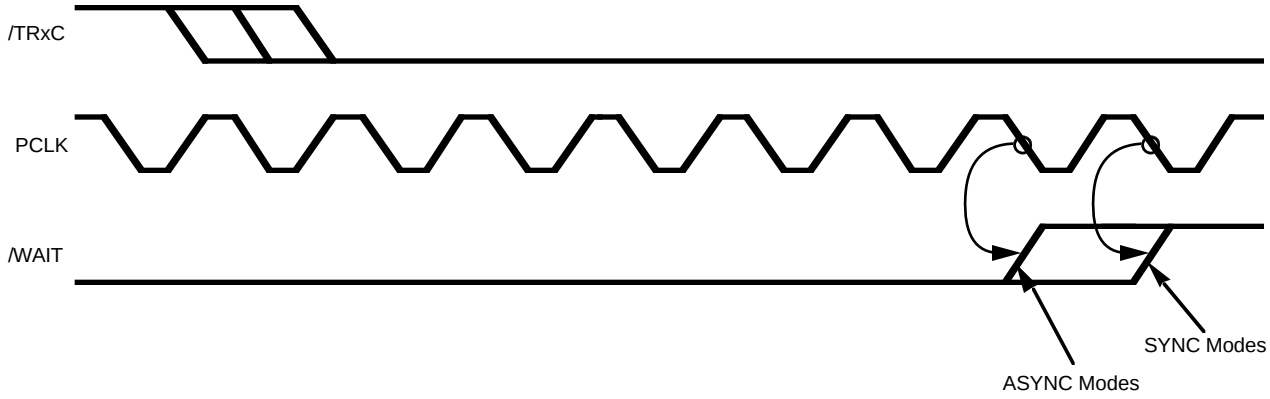



**Figure 2-25. Wait On Transmit Timing**

### 2.5.1.2 Wait On Receive

The Wait On Receive function is selected by setting D6 or WR1 to 0, D5 of WR1 to 1, and then enabling the function by setting D7 of WR1 to 1. In this mode, the /W//REQ pin carries the /WAIT signal, and is open-drain when inactive and Low when active. When the processor attempts to read data from the Receive FIFO when it is empty, the SCC asserts /WAIT until a character has reached the exit location of the FIFO (Figure 2-26).

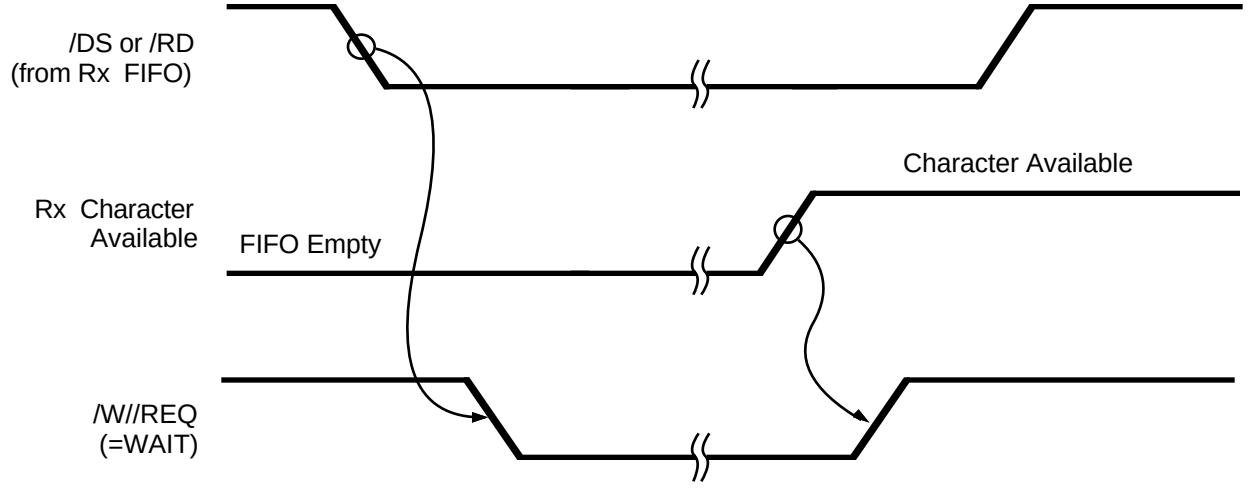



**Figure 2-26. Wait On Receive Timing**

This allows the use of a block move instruction to transfer the receive data. In the case of the Z80X30, /WAIT goes active in response to /DS going active, but only if RR8 is being accessed and a read is attempted. In all other cases, /WAIT remains open-drain. In the case of the Z85X30, /WAIT goes active in response to /RD going active, but only if the receive data FIFO is being accessed, either directly or via the pointers. The /WAIT pin is released in response to the falling edge of PCLK. Details of the timing are shown in Figure 2-27.

Care must be taken when this mode is used. The /WAIT pin stays active as long as the Receive FIFO remains empty. When the CPU access the SCC, the CPU remains in the wait state until data gets into the Receive FIFO, freezing the system.

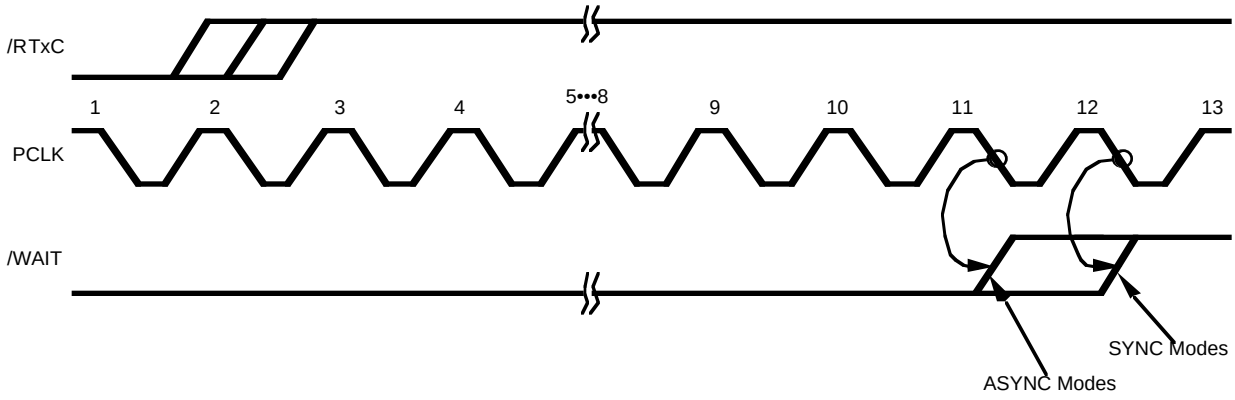



**Figure 2-27. Wait On Receive Timing**

## 2.5 BLOCK/DMA TRANSFER (Continued)

### 2.5.2 DMA Requests

The two DMA request pins /W//REQ and /DTR//REQ can be programmed for DMA requests. The /W//REQ pin is used as either a transmit or a receive request, and the /DTR//REQ pin can be used as a transmit request only. For full-duplex operation, the /W//REQ is used for receive, and the /DTR//REQ is used for transmit. These modes are described below.

#### 2.5.2.1 DMA Request on ESCC

Transmit DMA request is also affected by WR7' bit D5. As noted earlier, WR7' D5 affects both the transmit interrupt and DMA request generation similarly.

**Note:** WR7' D3 is ignored by the Receive Request function. This allows a DMA to transfer all bytes out of the Receive FIFO and still maintain the full advantage of the FIFO when the DMA has a long latency response acquiring the data bus.

Bit D5 of WR7' is set to 1 after reset to maintain maximum compatibility with SCC designs. This is necessary because if WR7' D5=0 when the request function is enabled, requests are made in rapid succession to fill the FIFO. Consequently, some designs which require an edge to be detected for each data transfer may not recover fast enough to detect the edges. This is handled by programming WR7' D5=1, or changing the DMA to be level sensitive instead of edge sensitive. Programming WR7' D5=0 has the advantage of the DMA requesting to keep the FIFO full. Therefore, if the CPU is busy, a significantly longer latency can be tolerated without the transmitter under-running.

#### 2.5.2.2 DMA Request On Transmit (using /W//REQ)

The Request On Transmit function is selected by setting D6 of WR1 to 1, D5 of WR1 to 0, and then enabling the function by setting D7 of WR1 to 1. In this mode, the /W//REQ pin carries the /REQ signal, which is active Low. When this mode is selected but not yet enabled, the /W//REQ is driven High.

The /REQ pin generates a falling edge for each byte written to the transmit buffer when the DMA controller is to write new data. For the Z80X30, the /REQ pin then goes inactive on the falling edge of the DS that writes the new data (see AC spec #26, TdDSf(REQ)) For the Z85X30, the /REQ pin then goes inactive on the falling edge of the WR strobe that writes the new data (see AC spec #33, TdWRf(REQ)) This is shown in Figure 2-28.

**Note:** The /REQ pin follows the state of the transmit buffer even though the transmitter is disabled. Thus, if the /REQ is enabled, the DMA writes data to the SCC before the transmitter is enabled. This will not cause a problem in Asynchronous mode, but it may cause problems in Synchronous mode because the SCC sends data in preference to flags or sync characters. It may also complicate the CRC initialization, which cannot be done until after the transmitter is enabled.

On the ESCC, this complication can be avoided in SDLC mode by using the Automatic SDLC Opening Flag Transmission feature and the Auto EOM reset feature, which also resets the transmit CRC (see Section 4.4.1 for details). Applications using other synchronous modes should enable the transmitter before enabling the /REQ function.

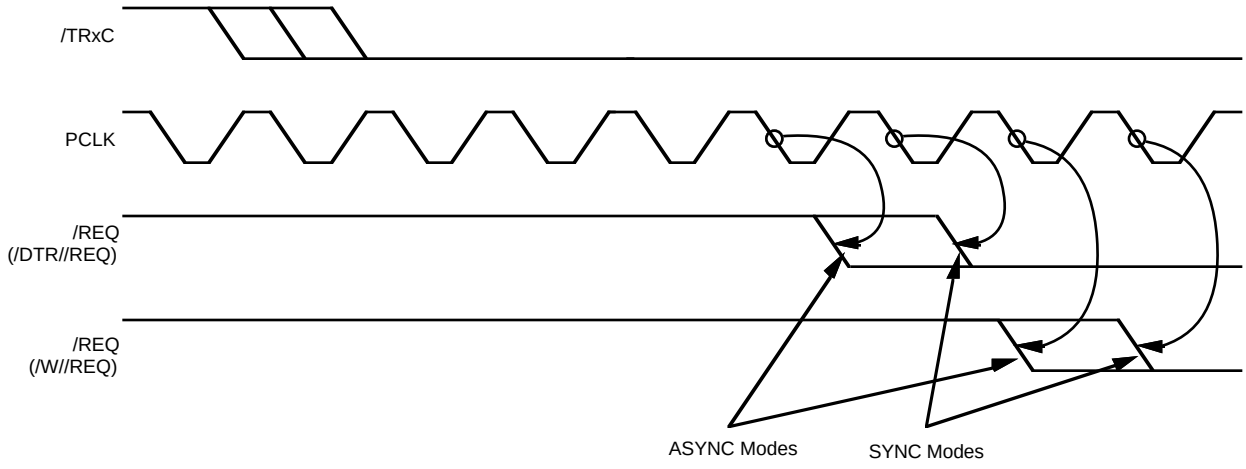



**Figure 2-28. Transmit Request Assertion**

With only one exception, the /REQ pin directly follows the state of the transmit buffer (for the ESCC as programmed by WR7' D5) in this mode. The SCC generates only one falling edge on /REQ per character requested and the timing for this is shown in Figure 2-29.

The one exception occurs in synchronous modes at the end of a CRC transmission. At the end of a CRC transmission, when the closing flag or sync character is loaded into the Transmit Shift Register, /REQ is pulsed High for one PCLK cycle. The DMA uses this falling edge on /REQ to write the first character of the next frame to the SCC. In the case of the Z80X30, /REQ goes High in response to the falling edge of DS, but only if the appropriate channel transmit buffer in the SCC is accessed. This is shown in Figure 2-25. In the case of the Z85X30, /REQ goes High in response to the falling edge of /WR, but only when the appropriate channel transmit buffer in the SCC is accessed. This is shown in Figure 2-30.

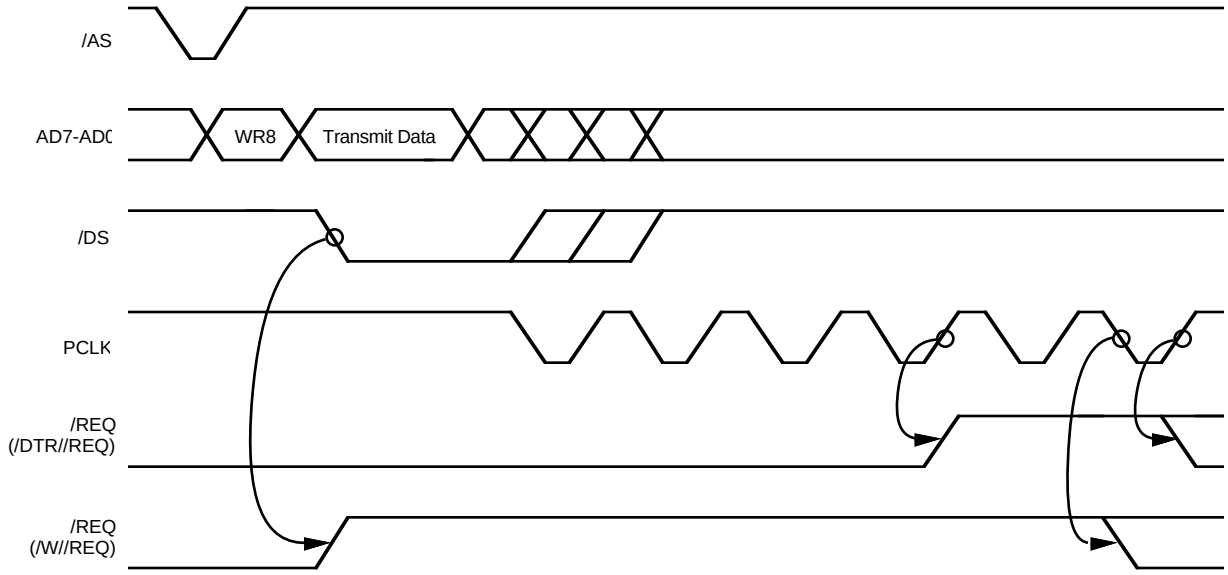



**Figure 2-29. Z80X30 Transmit Request Release**

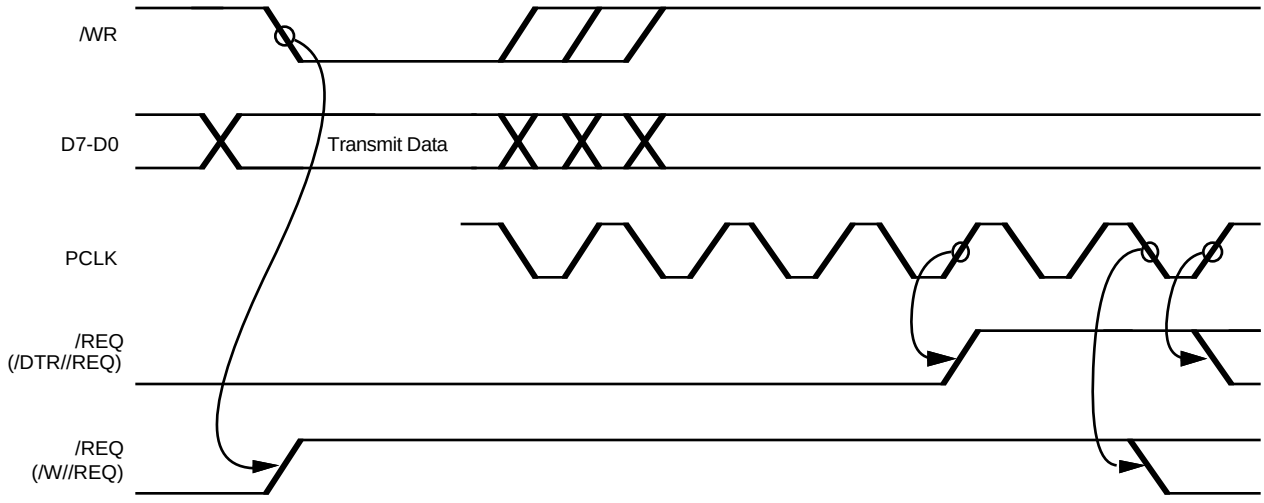



**Figure 2-30. Z85X30 Transmit Request Release**

## 2.5 BLOCK/DMA TRANSFER (Continued)

**2.5.2.3 DMA Request On Transmit (using /DTR//REQ)**
A second Request on Transmit function is available on the /DTR//REQ pin. This mode is selected by setting D2 of WR14 to 1. /REQ goes Low when the Transmit FIFO is empty if WR7' D5=1, or when the exit location of the Transmit FIFO is empty if WR7' D5=0. In the Request mode, /REQ follows the state of the Transmit FIFO even though the transmitter is disabled. While D2 of WR14 is set to 0, the /DTR//REQ pin is /DTR and follows the inverted state of D7 in WR5. This pin is High after a channel or hardware reset and in the DTR mode.

The /DTR//REQ pin goes inactive High between each transfer for a minimum of one PCLK cycle (Figure 2-31).

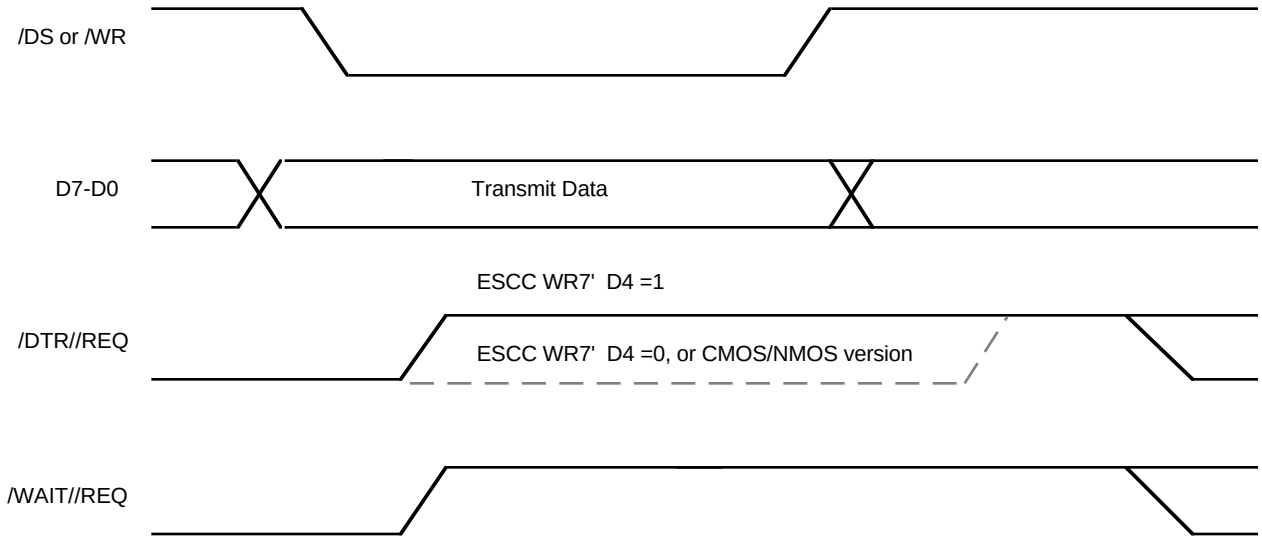



**Figure 2-31. /DTR//REQ Deassertion Timing**

***ESCC:***
***The timing of deactivation of this pin is programmable through WR7' bit D4. The /DTR//REQ waits until the write operation has been completed before going inactive. Refer to Z85230 AC spec #35a TdWRr(REQ) and Z80230 AC spec #27a TdDSr(REQ). This mode is compatible with the SCC and guarantees that any subsequent access to the ESCC does not violate the valid access recovery time requirement.***

***If WR7' D4=1, the /DTR//REQ is deactivated with identical timing as the /W/REQ pin. Refer to Z85230 AC spec #35b TdWRr(REQ) and Z80230 AC spec #27b TdDSr(REQ). This feature is beneficial to applications needing the DMA request to be deasserted quickly. It prevents a full Transmit FIFO from being overwritten due to the assertion of REQUEST being too long and being recognized as a request for more data.***

***Note: If WR7' D4=1, analysis should be done to verify that the ESCC is not repeatedly accessed in less than four PCLKs. However, since many DMAs require four clock cycles to transfer data, this typically is not a problem.***

In the Request mode, /REQ will follow the state of the transmit buffer even though the transmitter is disabled. Thus, if /REQ is enabled before the transmitter is enabled, the DMA may write data to the SCC before the transmitter is enabled. This does not cause a problem in Asynchronous mode, but may cause problems in Synchronous modes because the SCC sends data in preference to flags or sync characters. It may also complicate the CRC initialization, which cannot be done until after the transmitter is enabled. On the ESCC, this complication can be avoided in SDLC mode by using the Automatic SDLC Opening Flag Transmission feature and Auto EOM reset feature which also resets the transmit CRC. (See section 4.4.1.2 for details). Applications using other synchronous modes should enable the transmitter before enabling the /REQ function.

With only one exception, the /REQ pin directly follows the state of the Transmit FIFO (for ESCC, as programmed by WR7' D5) in this mode. The one exception occurs in synchronous modes at the end of a CRC transmission. At the end of a CRC transmission, when the closing flag or sync character is loaded into the Transmit Shift Register, /REQ is pulsed High for one PCLK cycle. The DMA uses this falling edge on /REQ to write the first character of the next frame to the SCC.

#### 2.5.2.4 DMA Request On Receive

The Request On Receive function is selected by setting D6 and D5 of WR1 to 1 and then enabling the function by setting D7 of WR1 to 1. In this mode, the /W//REQ pin carries the /REQ signal, which is active Low. When REQ on Receive is selected, but not yet enabled (WR1 D7=0), the /W//REQ pin is driven High. When the enable bit is set, /REQ goes Low if the Receive FIFO contains a character at the time, or will remain High until a character enters the Receive FIFO. Note that the /REQ pin follows the state of the Receive FIFO even though the receiver is disabled. Thus, if the receiver is disabled and /REQ is still enabled, the DMA transfers the previously received data correctly. In this mode, the /REQ pin directly follows the state of the Receive FIFO with only one exception. /REQ goes Low when a character enters the Receive FIFO and remains Low until this character is removed from the Receive FIFO.

The SCC generates only one falling edge on /REQ per character transfer requested (Figure 2-32). The one exception occurs in the case of a special receive condition in the Receive Interrupt on First Character or Special Condition mode, or the Receive Interrupt on Special Condition Only mode. In these two interrupt modes, any receive character with a special receive condition is locked at the top of the FIFO until an Error Reset command is issued. This character in the Receive FIFO would ordinarily cause additional DMA Requests after the first time it is read. However, the logic in the SCC guarantees only one falling edge on /REQ by holding /REQ High from the time the character with the special receive condition is read, and the FIFO locked, until after the Error Reset command has been issued.

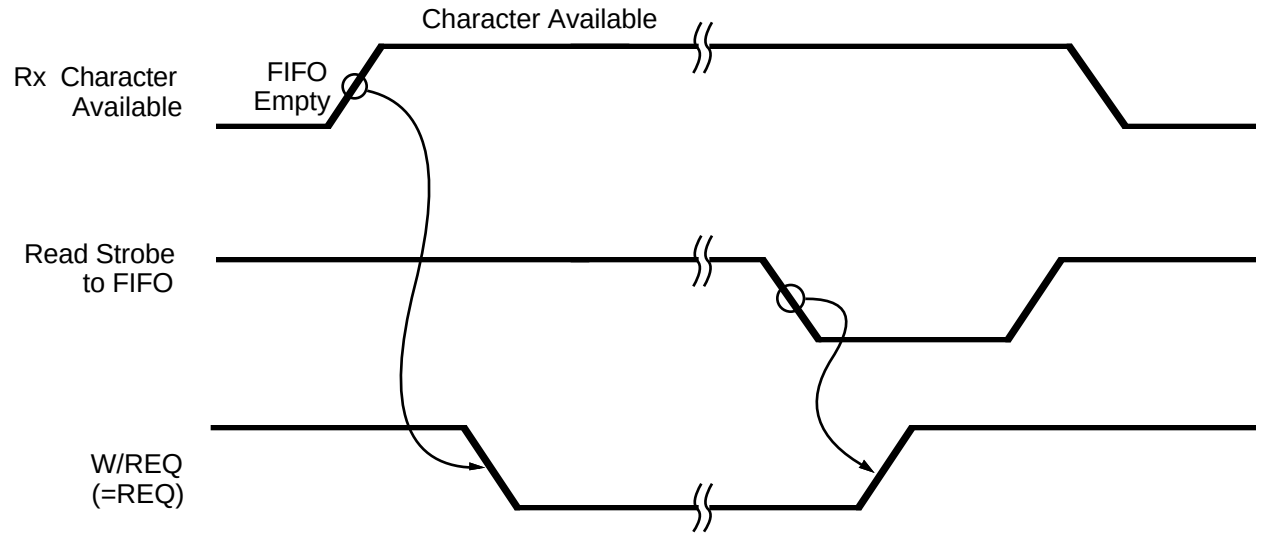



**Figure 2-32. DMA Receive Request Assertion**

## 2.5 BLOCK/DMA TRANSFER (Continued)

Once the FIFO is locked, it allows the checking of the Receive Error FIFO (RR1) to find the cause of the error. Locking the data FIFO, therefore, stops the error status from popping out of the Receive Error FIFO. Also, since the DMA request becomes inactive, the interrupt (Special Condition) is serviced.

Once the FIFO is unlocked by the Error Reset command, /REQ again follows the state of the receive buffer.
In the case of the Z80X30, /REQ goes High in response to the falling edge of /DS, but only if the appropriate receive buffer in the SCC is accessed (Figure 2-33). In the case of the Z85X30, /REQ goes High in response to the falling edge of /RD, but only when the appropriate receive buffer in the SCC is accessed (Figure 2-34).

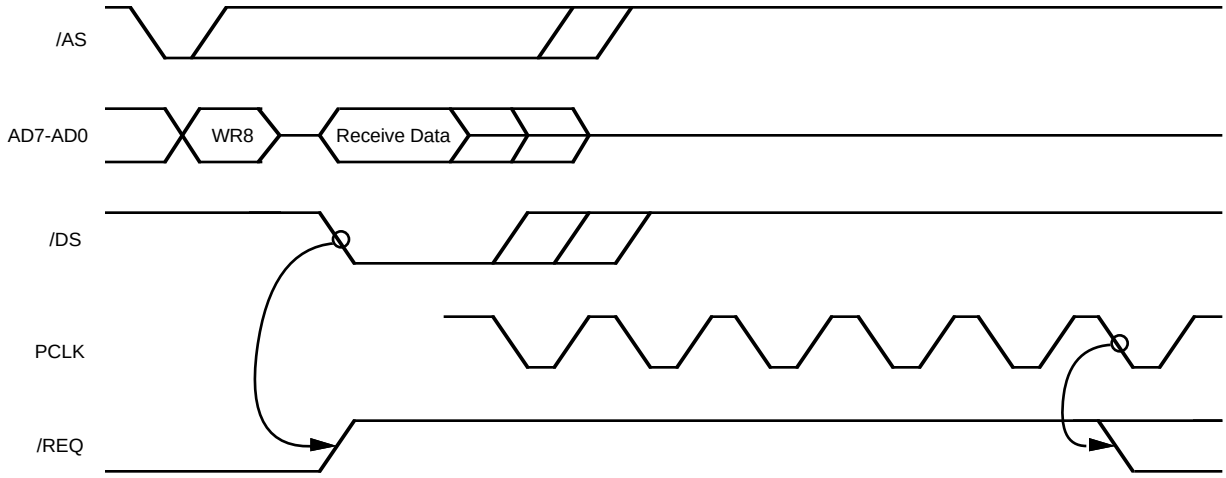



**Figure 2-33. Z80X30 Receive Request Release**

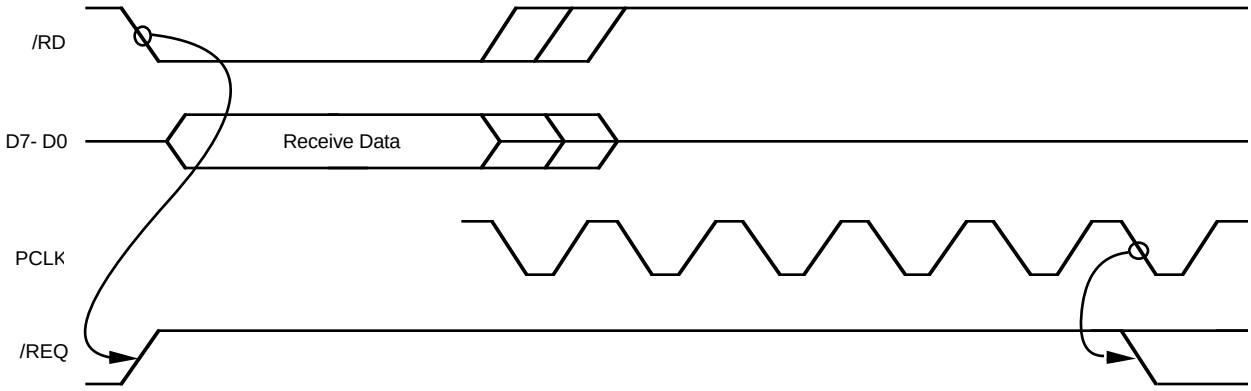



**Figure 2-34. Z85X30 Receive Request Release**

## 2.6 TEST FUNCTIONS

The SCC contains two other features useful for diagnostic purposes, controlled by bits in WR14. They are Local Loopback and Auto Echo.

### 2.6.1 Local Loopback

Local Loopback is selected when WR14 bit D4 is set to 1. In this mode, the output of the transmitter is internally connected to the input of the receiver. At the same time, the TxD pin remains connected to the transmitter. In this mode, the /DCD pin is ignored as a receive enable and the /CTS pin is ignored as a transmitter enable even if the Auto Enable mode has been selected. Note that the DPLL input is connected to the RxD pin, not to the input of the receiver. This precludes the use of the DPLL in Local Loopback. Local Loopback is shown schematically in Figure 2-35.

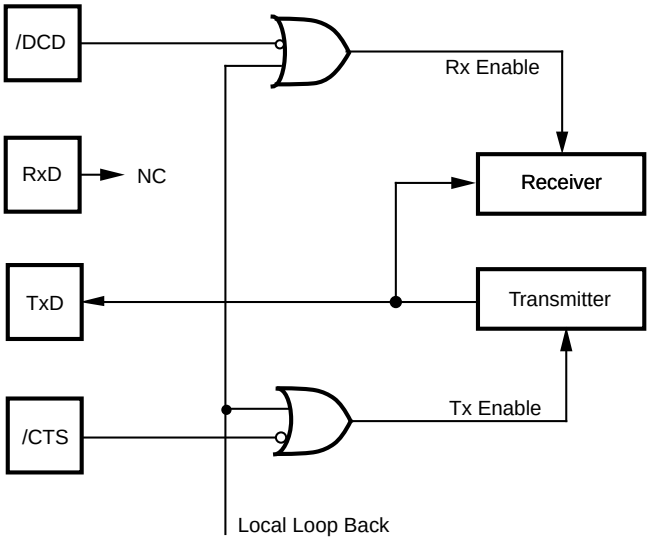



**Figure 2-35. Local Loopback**

### 2.6.2 Auto Echo

Auto Echo is selected when bit D3 of WR14 is set to 1. In this mode, the TxD pin is connected directly to the RxD pin, and the receiver input is connected to the RxD pin. In this mode, the /CTS pin is ignored as a transmitter enable and the output of the transmitter does not connect to anything. If both the Local Loopback and Auto Echo bits are set to 1, the Auto Echo mode is selected, but both the /CTS pin and /DCD pin are ignored as auto enables. This should not be considered a normal operating mode (Figure 2-36).

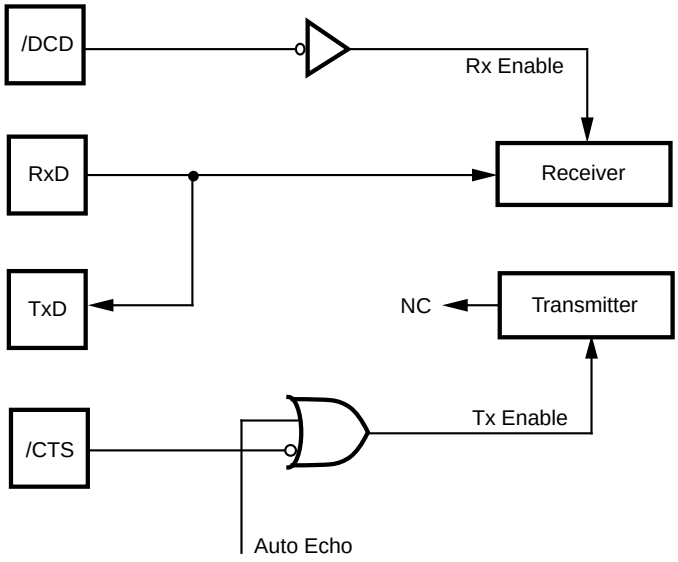



**Figure 2-36. Auto Echo**

# CHAPTER 3
## SCC/ESCC ANCILLARY SUPPORT CIRCUITRY

## 3.1 INTRODUCTION

The serial channels of the SCC are supported by ancillary circuitry for generating clocks and performing data encoding and decoding. This chapter presents a description of these functional blocks.

**Note to ESCC/CMOS Users:** The maximum input frequency to the DPLL has been specified as two times the PCLK frequency (Spec #16b TxRX(DPLL)). There are no changes to the baud rate generators from the NMOS to the CMOS/ESCC.

**Note to SCC Users:** The ancillary circuitry in the ESCC is the same as in the SCC with the following noted changes. The DPLL (Dual Phased-Locked Loop) output, when used as the transmit clock source, has been changed to be free of jitter. Consequently, this only affects the use of the DPLL as the transmit clock source (it is typically used for the receive clock source), this has no effect on using the DPLL as the receive clock source.

## 3.2 BAUD RATE GENERATOR

The Baud Rate Generator (BRG) is essential for asynchronous communications. Each channel in the SCC contains a programmable baud rate generator. Each generator consists of two 8-bit, time-constant registers forming a16-bit time constant, a 16-bit down counter, and a flip-flop on the output so that it outputs a square wave. On start-up, the flip-flop on the output is set High, so that it starts in a known state, the value in the time-constant register is loaded into the counter, and the counter begins counting down. When a count of zero is reached, the output of the baud rate generator toggles, the value in the time-constant register is loaded into the counter, and the process starts over. The programmed time constant is read from RR12 and RR13. A block diagram of the baud rate generator is shown in Figure 3-1.

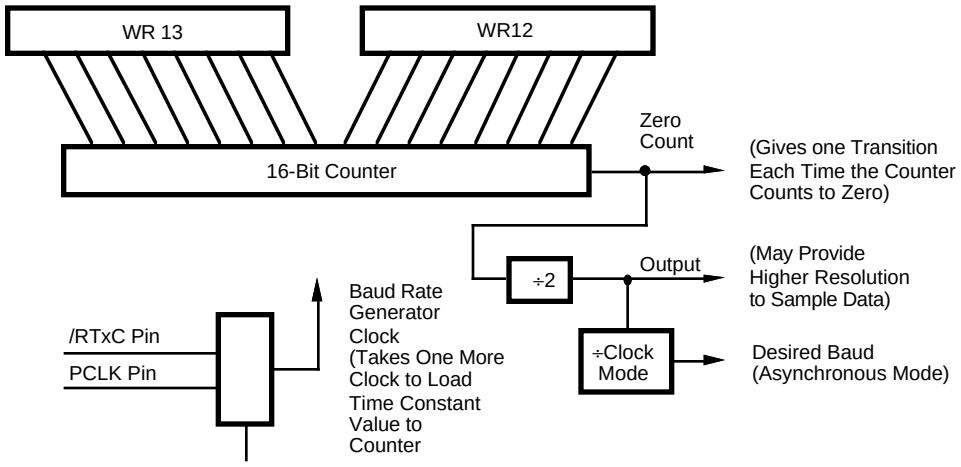



**Figure 3-1. Baud Rate Generator**

## 3.2 BAUD RATE GENERATOR (Continued)

The time-constant can be changed at any time, but the new value does not take effect until the next load of the counter (i.e., after zero count is reached).

No attempt is made to synchronize the loading of a new time-constant with the clock used to drive the generator. When the time-constant is to be changed, the generator should be stopped first by writing WR14 D0=0. After loading the new time constant, the BRG can be started again. This ensures the loading of a correct time constant, but loading does not take place until zero count or a reset occurs.

If neither the transmit clock nor the receive clock are programmed to come from the /TRxC pin, the output of the baud rate generator may be made available for external use on the /TRxC pin.

**Note:** This feature is very useful for diagnostic purposes. By programming the output of the baud rate generator as output on the /TRxC pin, the BRG is source and time tested, and the programmed time constant verified.

The clock source for the baud rate generator is selected by bit D1 of WR14. When this bit is set to 0, the BRG uses the signal on the /RTxC pin as its clock, independent of whether the /RTxC pin is a simple input or part of the crystal oscillator circuit. When this bit is set to 1, the BRG is clocked by the PCLK. To avoid metastable problems in the counter, this bit should be changed only while the baud rate generator is disabled, since arbitrarily narrow pulses can be generated at the output of the multiplexer when it changes status.

The BRG is enabled while bit D0 of WR14 is set to 1. It is disabled while WR14 D0=0 and after a hardware reset (but not a software reset). To prevent metastable problems when the baud rate generator is first enabled, the enable bit is synchronized to the baud rate generator clock. This introduces an additional delay when the baud rate generator is first enabled (Figure 3-2). The baud rate generator is disabled immediately when bit D0 of WR14 is set to 0, because the delay is only necessary on start-up. The baud rate generator is enabled and disabled on the fly, but this delay on start-up must be taken into consideration.

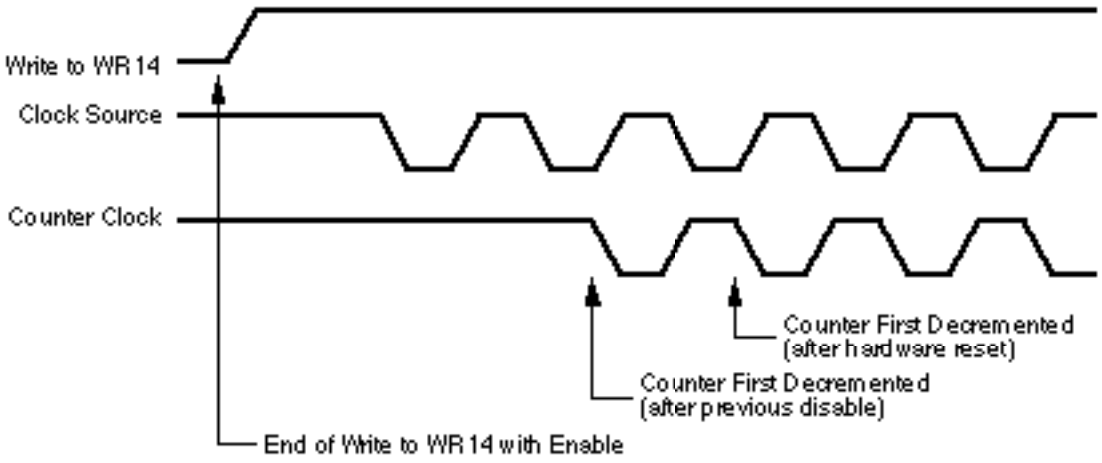



**Figure 3-2. Baud Rate Generator Start Up**

The formulas relating the baud rate to the time-constant and vice versa are shown below.

$$\text{Time Constant} = \frac{\text{Clock Frequency}}{2 \times (\text{Clock Mode}) \times (\text{Baud Rate})} - 2$$

$$\text{Baud Rate} = \frac{\text{Clock Frequency}}{2 \times (\text{Clock Mode}) \times (\text{Time Constant} + 2)}$$

In these formulas, the BRG clock frequency (PCLK or /RTxC) is in Hertz, the desired baud rate in bits/sec, Clock Mode is 1 in sync modes, 1, 16, 32 or 64 in async mode and the time constant is dimensionless. The example in Table 3-1 assumes a 2.4576 MHz clock (from /RTxC) factor of 16 and shows the time constant for a number of popular baud rates.

For example:

$$TC = \frac{2.4576 \times 10^6}{(2 \times 16) \times 150} - 2 = 510$$

.

**Table 3-1. Baud Rates for 2.4576 MHz Clock and 16x Clock Factor**

| Baud Rate | Time Constant | |
|---|---|---|
| | Decimal | Hex |
| 38400 | 0 | 0000 |
| 19200 | 2 | 0002 |
| 9600 | 6 | 0006 |
| 4800 | 14 | 000E |
| 2400 | 30 | 001E |
| 1200 | 62 | 003E |
| 600 | 126 | 007E |
| 300 | 254 | 00FE |
| 150 | 510 | 01FE |

Other commonly used clock frequencies include 3.6846, 4.6080, 4.91520, 6.144, 7.3728, 9.216, 9.8304, 12.288, 14.7456, 19.6608 (units in MHz).

Initializing the BRG is done in three steps. First, the time-constant is determined and loaded into WR12 and WR13. Next, the processor must select the clock source for the BRG by setting bit D1 of WR14. Finally, the BRG is enabled by setting bit D0 of WR14 to 1.

**Note:** The first write to WR14 is not necessary after a hardware reset if the clock source is the /RTxC pin. This is because a hardware reset automatically selects the /RTxC pin as the BRG clock source.

## 3.3 DATA ENCODING/DECODING

Data encoding is utilized to allow the transmission of clock and data information over the same medium. This saves the need to transmit clock and data over separate medium as would normally be required for synchronous data. The SCC provides four different data encoding methods, selected by bits D6 and D5 in WR10. An example of these four encoding methods is shown in Figure 3-3. Any encoding method is used in any X1 mode in the SCC, asynchronous or synchronous. The data encoding selected is active even though the transmitter or receiver is idling or disabled.

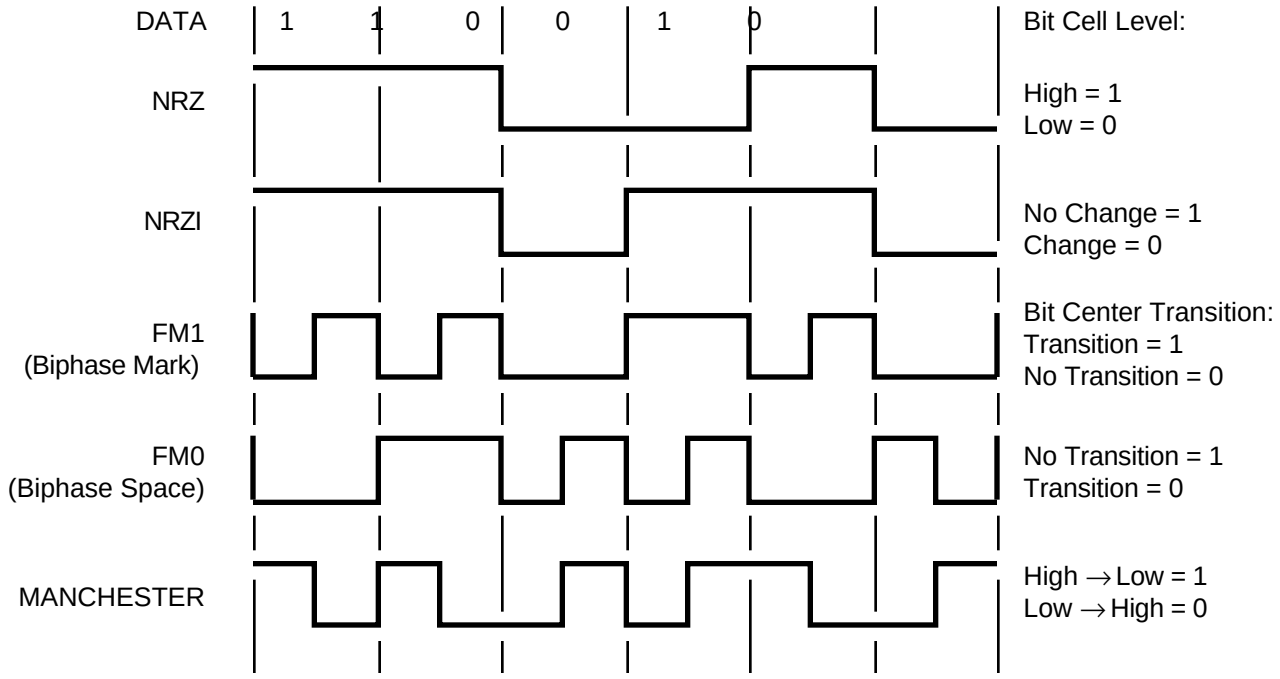



**Figure 3-3. Data Encoding Methods**

**NRZ (Non-Return to Zero).** In NRZ, encoding a 1 is represented by a High level and a 0 is represented by a Low level. In this encoding method, only a minimal amount of clocking information is available in the data stream in the form of transitions on bit-cell boundaries. In an arbitrary data pattern, this may not be sufficient to generate a clock for the data from the data itself.

**NRZI (Non-Return to Zero Inverted).** In NRZI, encoding a 1 is represented by no change in the level and a 0 is represented by a change in the level. As in NRZ, only a minimal amount of clocking information is available in the data stream, in the form of transitions on bit cell boundaries. In an arbitrary data pattern this may not be sufficient to generate a clock for the data from the data itself. In the case of SDLC, where the number of consecutive 1s in the data stream is limited, a minimum number of transitions to generate a clock are guaranteed.

***ESCC:***
***TxD Pin Forced High in SDLC feature. When the ESCC is programmed for SDLC mode with NRZI data encoding and mark idle (WR10 D6=0, D5=1, D3=1), the TxD pin is automatically forced high when the transmitter goes to the mark idle state. There are several different ways for the transmitter to go into the idle state. In each of the following cases the TxD pin is forced high when the mark idle condition is reached: data, CRC, flag and idle; data, flag and idle; data, abort (on underrun) and idle; data, abort (command) and idle; idle flag and command to idle mark. The Force High feature is disabled when the mark idle bit is reset. The TxD pin is forced High on the falling edge of the TxC cycle after the falling edge of the last bit of the closing flag. Using SDLC Loop mode is independent of this feature.***

***This feature is used in combination with the automatic SDLC opening flag transmission feature, WR7' D0=1, to assure that data packets are properly formatted. Therefore, when these features are used together, it is not necessary for the CPU to issue any commands when using the force idle mode in combination with NRZI data encoding. If WR7' D0 is reset, like the SCC, it is necessary to reset the mark idle bit (WR10 D2) to enable flag transmission before an SDLC packet is transmitted.***

**FM1 (Bi-phase Mark).** In FM1 encoding, also known as bi-phase mark, a transition is present on every bit cell boundary, and an additional transition may be present in the middle of the bit cell. In FM1, a 0 is sent as no transition in the center of the bit cell and a 1 is sent as a transition in the center of the bit cell. FM1 encoded data contains sufficient information to recover a clock from the data.

**FM0 (Bi-phase Space).** In FM0 encoding, also known as bi-phase space, a transition is present on every bit cell boundary and an additional transition may be present in the middle of the bit cell. In FM0, a 1 is sent as no transition in the center of the bit cell and a 0 is sent as a transition in the center of the bit cell. FM0 encoded data contains sufficient information to recover a clock from the data.

**Manchester (Bi-phase Level).** Manchester (bi-phase level) encoding always produces a transition at the center of the bit cell. If the transition is Low to High, the bit is 0. If the transition is High to Low, the bit is 1. Encoding of Manchester format requires an external circuit consisting of a 'D' flip-flop and four gates (Figure 3-4). The SCC is used to decode Manchester data by using the DPLL in the FM mode and programming the receiver for NRZ data (See Section 3.1.3).

**Data Encoding Initialization.** The data encoding method is selected in the initialization procedure before the transmitter and receiver are enabled, but no other restrictions apply. Note that in NRZ and NRZI, the receiver samples the data only on one edge, as shown in Figure 3-3. However, in FM1 and FM0, the receiver samples the data on both edges. Also, as shown in Figure 3-3, the transmitter defines bit cell boundaries by one edge in all cases and uses the other edge in FM1 and FM0 to create the mid-bit transition.

## 3.3 DATA ENCODING/DECODING (Continued)

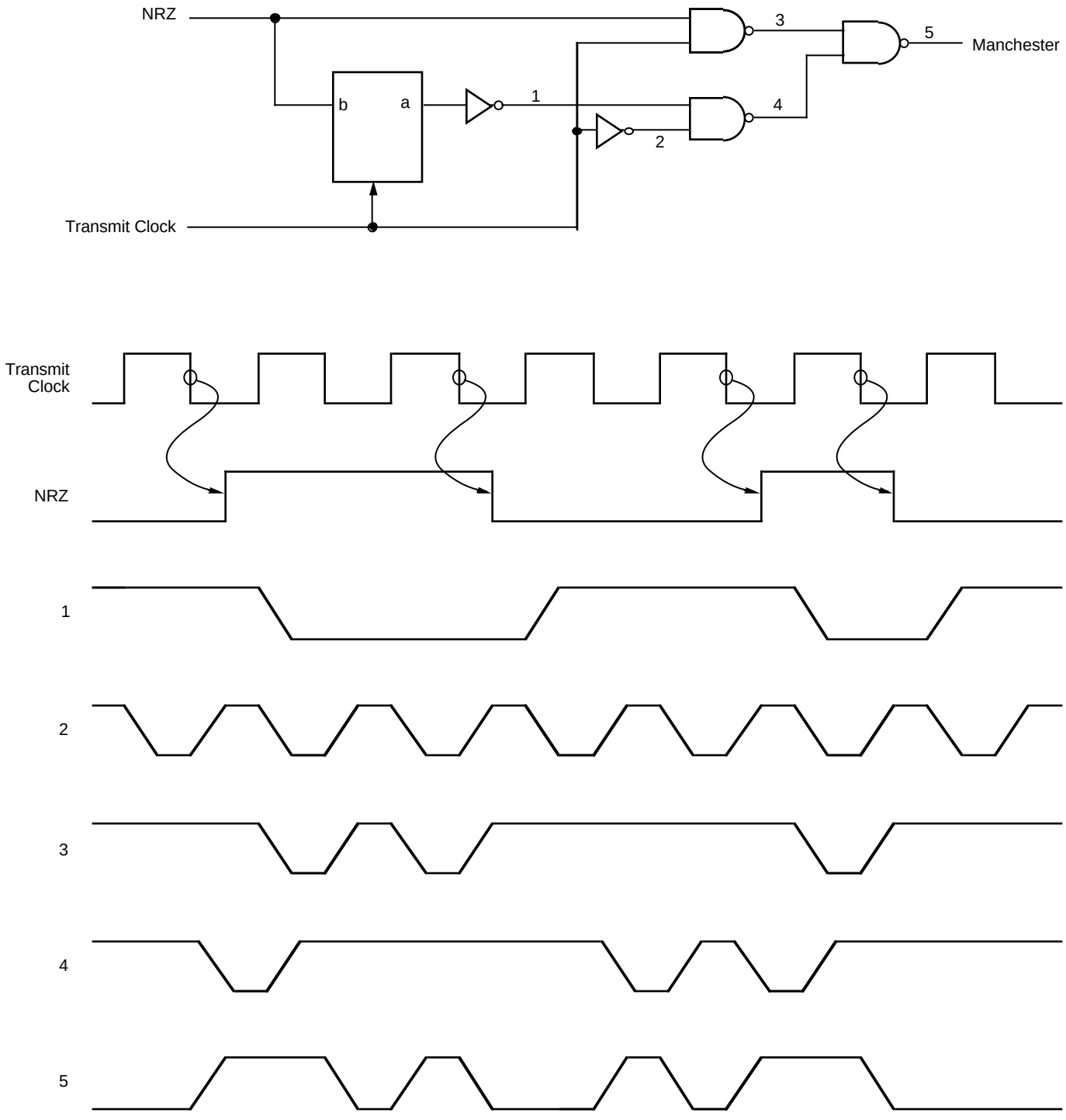



**Figure 3-4. Manchester Encoding Circuit**

## 3.4 DPLL DIGITAL PHASE-LOCKED LOOP

Each channel of the SCC contains a digital phase-locked loop that can be used to recover clock information from a data stream with NRZI, FM, NRZ, or Manchester encoding. The DPLL is driven by a clock nominally at 32 (NRZI) or 16 (FM) times the data rate. The DPLL uses this clock, along with the data stream, to construct a receive clock for the data. This clock can then be used as the SCC receive clock, the transmit clock, or both.

Figure 3-5 shows a block diagram of the digital phase-locked loop. It consists of a 5-bit counter, an edge detector, and a pair of output decoders. The clock for the DPLL comes from the output of a two-input multiplexer, and the two outputs go to the transmitter and receive clock multiplexers. The DPLL is controlled by seven commands encoded in WR14 bits D7, D6 and D5.

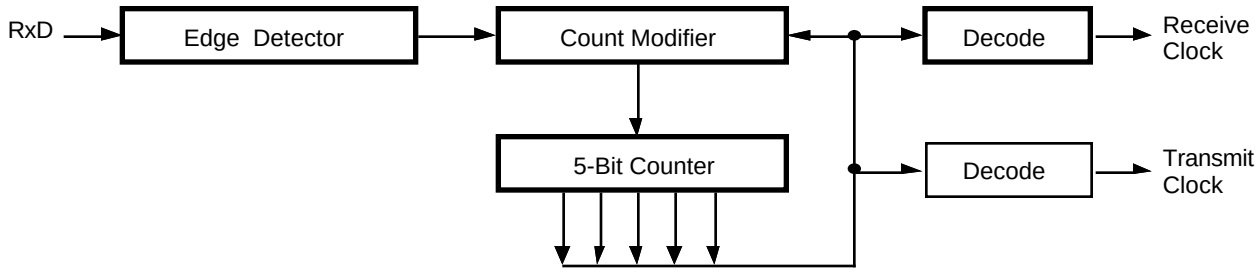



**Figure 3-5. Digital Phase-Locked Loop**

The clock source for the DPLL is selected issuing one of the two commands in WR14, that is:

WR14 (7-5) = 100 selects the BRG
WR14 (7-5) = 101 selects the /RTxC pin

The first command selects the baud rate generator as the clock source. The other command selects the /RTxC pin as the clock source, independent of whether the /RTxC pin is a simple input or part of the crystal oscillator circuit.

Initialization of the DPLL is done at any time during the initialization sequence, but should be done after the clock modes have been selected in WR11, and before the receiver and transmitter are enabled. When initializing the DPLL, the clock source should be selected first, followed by the selection of the operating mode.

To avoid metastable problems in the counter, the clock source selection is made only while DPLL is disabled, since arbitrarily narrow pulses are generated at the output of the multiplexer when it changes status.

The DPLL is programmed to operate in one of two modes, as selected by commands in WR14.

WR14 (7-5) = 111 selects NRZI mode
WR14 (7-5) = 110 selects FM mode

**Note:** A channel or hardware reset disables the DPLL, selects the /RTxC pin as the clock source for the DPLL, and places it in the NRZI mode.

As in the case of the clock source selection, the mode of operation is only changed while the DPLL is disabled to prevent unpredictable results.

In the NRZI mode, the DPLL clock must be 32 times the data rate. In this mode, the transmit and receive clock outputs of the DPLL are identical, and the clocks are phased so that the receiver samples the data in the middle of the bit cell. In NRZI mode, the DPLL does not require a transition in every bit cell, so this mode is useful for recovering the clocking information from NRZ and NRZI data streams.

In the FM mode, the DPLL clock must be 16 times the data rate. In this mode, the transmit clock output of the DPLL lags the receive clock outputs by 90 degrees to make the transmit and receive bit cell boundaries the same, because the receiver must sample FM data at one-quarter and three-quarters bit time.

The DPLL is enabled by issuing the Enter Search Mode command in WR14; that is WR14 (7-5) = 001. The Enter Search Mode command unlocks the counter, which is held while the DPLL is disabled, and enables the edge detector. If the DPLL is already enabled when this command is issued, the DPLL also enters Search Mode.

## 3.4 DPLL DIGITAL PHASE-LOCKED LOOP (Continued)

### 3.4.1 DPLL Operation in the NRZI Mode

To operate in NRZI mode, the DPLL must be supplied with a clock that is 32 times the data rate. The DPLL uses this clock, along with the receive data, to construct receive and transmit clock outputs that are phased to properly receive and transmit data.

To do this, the DPLL divides each bit cell into four regions, and makes an adjustment to the count cycle of the 5-bit counter dependent upon the region a transition on the receive data input occurred (Figure 3-6).

Ordinarily, a bit-cell boundary occurs between count 15 and count 16, and the DPLL output causes the data to be sampled in the middle of the bit cell. However, four different situations can occur:

If the bit-cell boundary (from space to mark) occurs anywhere during the second half of count 15 or the first half of count 16, the DPLL allows the transition without making a correction to its count cycle.

If the bit cell boundary (from space to mark) occurs between the middle of count 16 and count 31, the DPLL is sampling the data too early in the bit cell. In response to this, the DPLL extends its count by one during the next 0 to 31 counting cycle, which effectively moves the edge of the clock that samples the receive data closer to the center of the bit cell.

If the transition occurs between count 0 and the middle of count 15, the output of the DPLL is sampling the data too late in the bit cell. To correct this, the DPLL shortens its count by one during the next 0 to 31 counting cycle, which effectively moves the edge of the clock that samples the receive data closer to the center of the bit cell.

If the DPLL does not see any transition during a counting cycle, no adjustment is made in the following counting cycle.

If an adjustment to the counting cycle is necessary, the DPLL modifies count 5, either deleting it or doubling it. Thus, only the Low time of the DPLL output is lengthened or shortened.

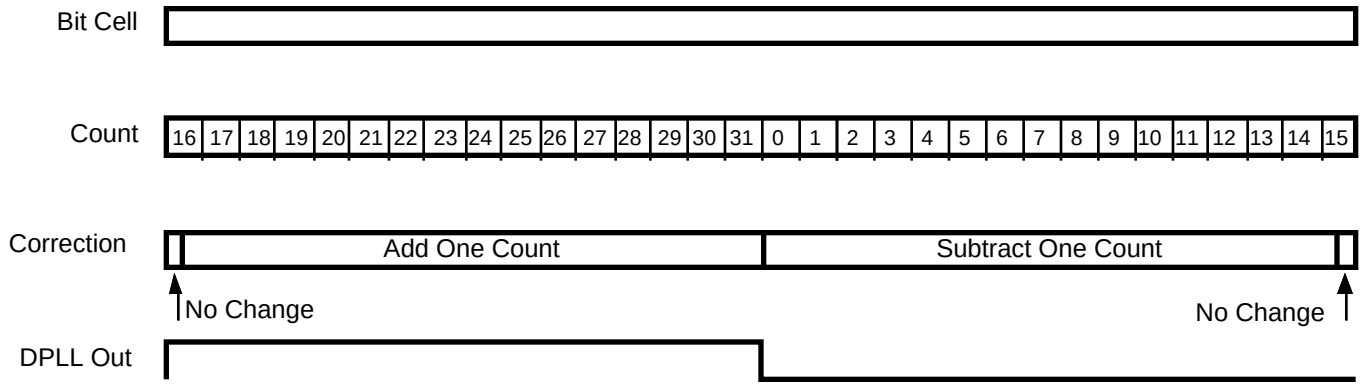



**Figure 3-6. DPLL in NRZI Mode**

While the DPLL is in search mode, the counter remains at count 16, where the DPLL outputs are both High. The missing clock latches in the DPLL, which may be accessed in RR10, are not used in NRZI mode. An example of the DPLL in operation is shown in Figure 3-7.

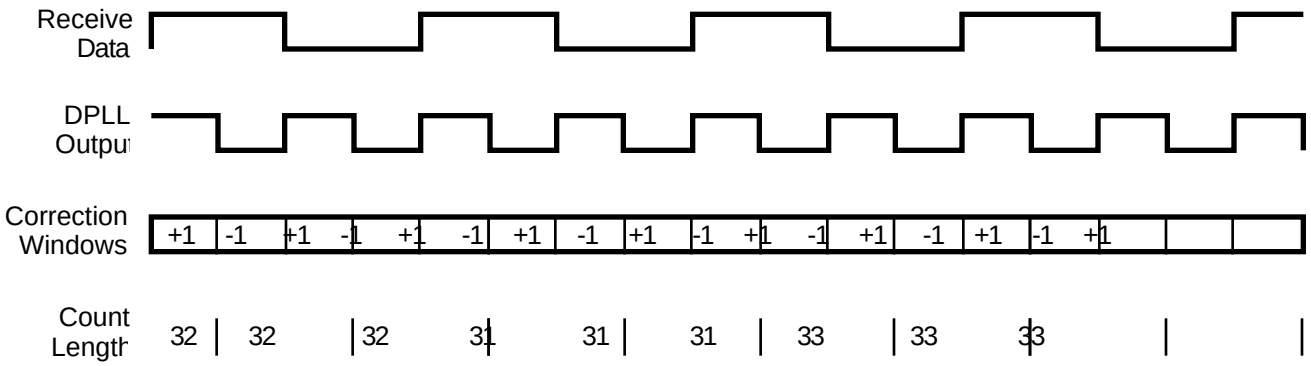



**Figure 3-7. DPLL Operating Example (NRZI Mode)**

## 3.4.2 DPLL Operation in the FM Modes

To operate in FM mode, the DPLL must be supplied with a clock that is 16 times the data rate. The DPLL uses this clock, along with the receive data, to construct, receive, and transmit clock outputs that are phased to receive and transmit data properly.

In FM mode, the counter in the DPLL counts from 0 to 31, but now each cycle corresponds to 2-bit cells. To make adjustments to remain in phase with the receive data, the DPLL divides a pair of bit cells into five regions, making the adjustment to the counter dependent upon which region the transition on the receive data input occurred (Figure 3-8).

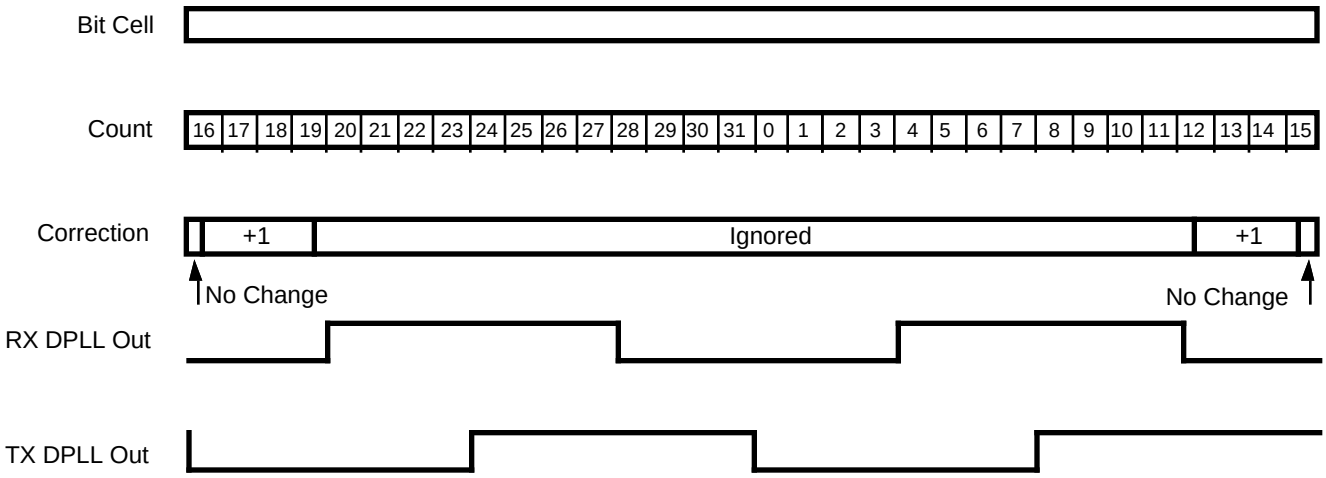



**Figure 3-8. DPLL Operation in the FM Mode**

## 3.4 DPLL DIGITAL PHASE-LOCKED LOOP (Continued)

In FM mode, the transmit clock and receive clock outputs from the DPLL are not in phase. This is necessary to make the transmit and receive bit cell boundaries coincide, since the receive clock must sample the data one-fourth and three-fourths of the way through the bit cell.

Ordinarily, a bit cell boundary occurs between count 15 or count 16, and the DPLL receive output causes the data to be sampled at one-fourth and three-fourths of the way through the bit cell.

However, four variations can occur:

If the bit-cell boundary (from space to mark) occurs anywhere during the second half of count 15 or the first half of count 16, the DPLL allows the transition without making a correction to its count cycle.

If the bit-cell boundary (from space to mark) occurs between the middle of count 16 and the middle of count 19, the DPLL is sampling the data too early in the bit cell. In response to this, the DPLL extends its count by one during the next 0 to 31 counting cycle, which effectively moves the receive clock edges closer to where they should be.

Any transitions occurring between the middle of count 19 in one cycle and the middle of count 12 during the next cycle are ignored by the DPLL. This guarantees that any data transitions in the bit cells do not cause an adjustment to the counting cycle.

If no transition occurs between the middle of count 12 and the middle of count 19, the DPLL is probably not locked onto the data properly. When the DPLL misses an edge, the One Clock Missing bit is RR10, it is set to 1 and latched. It will hold this value until a Reset Missing Clock command is issued in WR14, or until the DPLL is disabled or programmed to enter the Search mode. Upon missing this one edge, the DPLL takes no other action and does not modify its count during the next counting cycle.

If the DPLL does not see an edge between the middle of count 12 and the middle of count 19 in two successive 0 to 31 count cycles, a line error condition is assumed. If this occurs, the Two Clocks Missing bit in RR10 is set to 1 and latched. At the same time, the DPLL enters the Search mode. The DPLL makes the decision to enter the Search mode during count 2, where both the receive clock and transmit clock outputs are Low. This prevents any glitches on the clock outputs when the Search mode is entered. While in the Search mode, no clock outputs are provided by the DPLL. The Two Clocks Missing bit in RR10 is latched until a Reset Missing Clock command is issued in WR14, or until the DPLL is disabled or programmed to enter the Search mode.

While the DPLL is disabled, the transmit clock output of the DPLL may be toggled by alternately selecting FM and NRZI mode in the DPLL. The same is true of the receive clock.

While the DPLL is in the Search mode, the counter remains at count 16 where the receive output is Low and the transmit output is Low. This fact is used to provide a transmit clock under software control since the DPLL is in the Search mode while it is disabled.

As in NRZI mode, if an adjustment to the counting cycle is necessary, the DPLL modifies count 5, either deleting it or doubling it. If no adjustment is necessary, the count sequence proceeds normally.

When the DPLL is programmed to enter Search mode, only clock transitions should exist on the receive data pin. If this is not the case, the DPLL may attempt to lock on to the data transitions. If the DPLL does lock on to the data transitions, then the Missing Clock condition will inevitably occur because data transitions are not guaranteed every bit cell.

To lock in the DPLL properly, FM0 encoding requires continuous 1s received when leaving the Search mode. In FM1 encoding, continuous 0s are required; with Manchester encoded data this means alternating 1s and 0s. With all three of these data encoding methods there is always at least one transition in every bit cell, and in FM mode the DPLL is designed to expect this transition.

### 3.4.3 DPLL Operation in the Manchester Mode

The SCC can be used to decode Manchester data by using the DPLL in the FM mode and programming the receiver for NRZ data. Manchester encoded data contains a transition at the center of every bit cell; it is the direction of this transition that distinguishes a 1 from a 0. Hence, for Manchester data, the DPLL should be in FM mode (WR14 command D7=1, D6=1, D5=0), but the receiver should be set up to accept NRZ data (WR10 D6=0, D5=0).

### 3.4.4 Transmit Clock Counter (ESCC only)

The ESCC includes a Transmit Clock Counter which parallels the DPLL. This counter provides a jitter-free clock source to the transmitter by dividing the DPLL clock source by the appropriate value for the programmed data encoding format as shown in Figure 3-9. Therefore, in FM mode (FM0 or FM1), the counter output is the input frequency divided by 16. In NRZI mode, the counter frequency is the input divided by 32. The counter output replaces the DPLL transmit clock output, available as the transmit clock source. This has no effect on the use of the DPLL as the receive clock source.

The output of the transmit clock derived from this counter is available to the /TRxC pin when the DPLL output is selected as the transmit clock source. Care must be taken using ESCC in SDLC Loop mode with the DPLL. The SDLC Loop mode requires synchronized Tx and Rx clocks, but the ESCC's DPLL might be off-sync because of this Transmit Clock Counter. In SDLC Loop, one should instead echo the signal of the RxDPLL out to clock the receiver and transmitter to achieve synchronization. This can be programmed via bits D1-D0 in WR11.

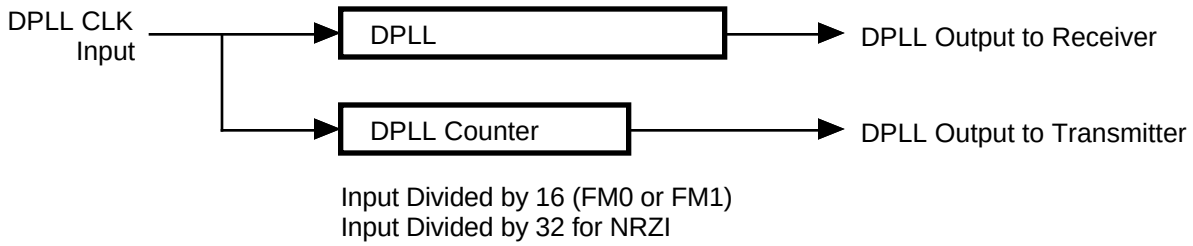



**Figure 3-9. DPLL Transmit Clock Counter Output (ESCC only)**

## 3.5 CLOCK SELECTION

The SCC can select several clock sources for internal and external use. Write Register 11 is the Clock Mode Control register for both the receive and transmit clocks. It determines the type of signal on the /SYNC and /RTxC pins and the direction of the /TRxC pin.

The SCC is programmed to select one of several sources to provide the transmit and receive clocks.

The source of the receive clock is controlled by bits D6 and D5 of WR11. The receive clock may be programmed to come from the /RTxC pin, the /TRxC pin, the output of the baud rate generator, or the receive output of the DPLL.

The source of the transmit clock is controlled by bits D4 and D3 of WR11. The transmit clock may be programmed to come from the /RTxC pin, the /TRxC pin, the output of the baud rate generator, or the transmit output of the DPLL.

Ordinarily, the /TRxC pin is an input, but it can become an output if this pin has not been selected as the source for the transmitter or the receiver, and bit D2 of WR11 is set to 1. The selection of the signal provided on the /TRxC output pin is controlled by bits D1 and D0 of WR11. The /TRxC pin is programmed to provide the output of the crystal oscillator, the output of the baud rate generator, the receive output of the DPLL or the actual transmit clock. If the output of the crystal oscillator is selected, but the crystal oscillator has not been enabled, the /TRxC pin is driven High. The option of placing the transmit clock signal on the /TRxC pin when it is an output allows access to the transmit output of the DPLL.

Figure 3-10 shows a simplified schematic diagram of the circuitry used in the clock multiplexing. It shows the inputs to the multiplexer section, as well as the various signal inversions that occur in the paths to the outputs.

Selection of the clocking options may be done anywhere in the initialization sequence, but the final values must be selected before the receiver, transmitter, baud rate generator, or DPLL are enabled to prevent problems from arbitrarily narrow clock signals out of the multiplexers. The same is true of the crystal oscillator, in that the output should be allowed to stabilize before it is used as a clock source.

Also shown are the edges used by the receiver, transmitter, baud rate generator and DPLL to sample or send data or otherwise change state. For example, the receiver samples data on the falling edge, but since there is an inversion in the clock path between the /RTxC pin and the receiver, a rising edge of the /RTxC pin samples the data for the receiver.

The following shows three examples for selecting different clocking options. Figure 3-11 shows the clock set up for asynchronous transmission, 16x clock mode using the on-chip oscillator with an external crystal. This example uses the oscillator as the input to the baud rate generator, although it can be used directly as the transmit or receive clock source. The registers involved are WR11 through WR14 and the figure shows the programming in these registers.

An example of asynchronous communication where a 1x clock is obtained from an external MODEM is shown in Figure 3-12. The data encoding is NRZ. Note that:

1. The BRG is not used under this configuration.

## 3.5 CLOCK SELECTION (Continued)

2. The x1 mode in Asynchronous mode is a combination of both synchronous and asynchronous transmission. The data is clocked by a common timing base, but characters are still framed with Start and Stop bits. Because the receiver waits for one clock period after detecting the first High-to-Low transition before beginning to assemble characters, the data and clock is synchronized externally. The x1 mode is the only mode in which a data encoding method other than NRZ is used.

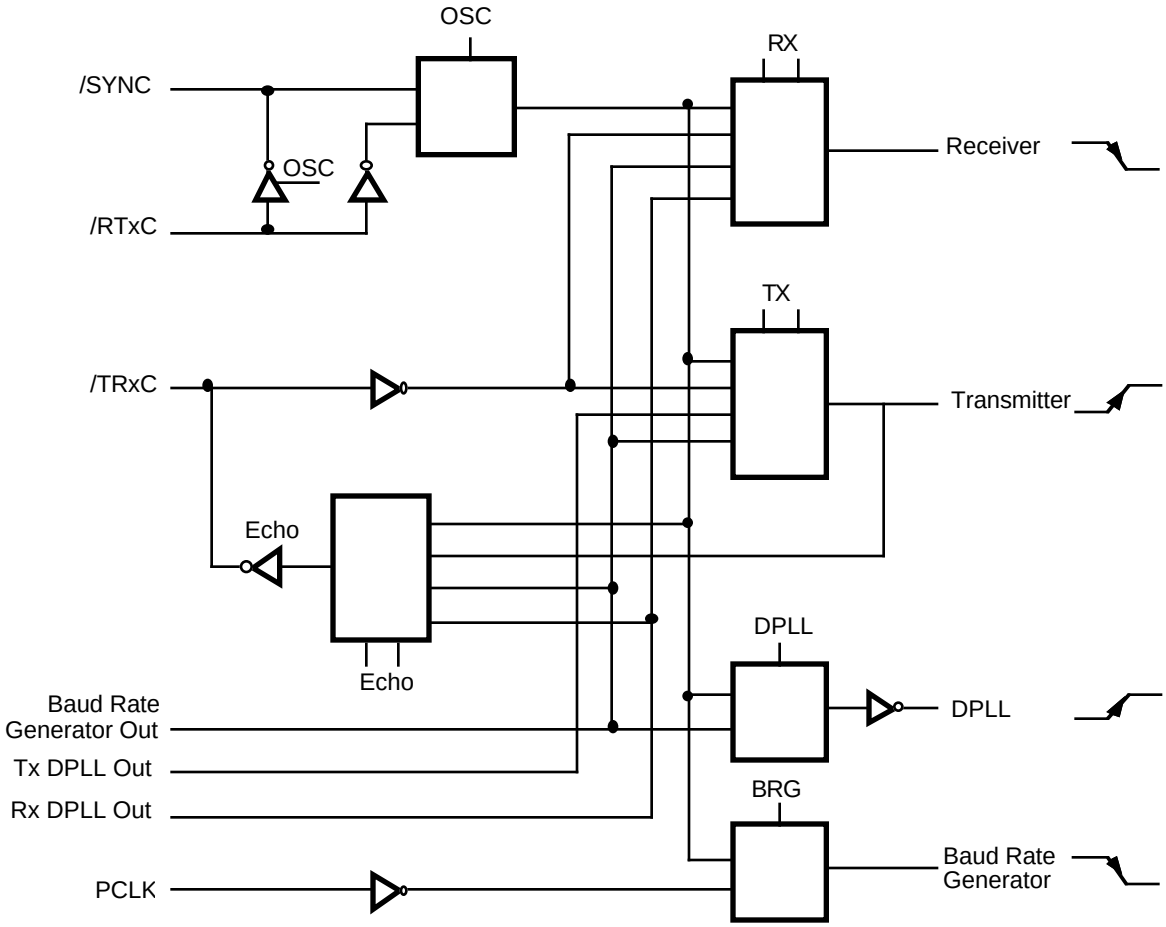



**Figure 3-10. Clock Multiplexer**

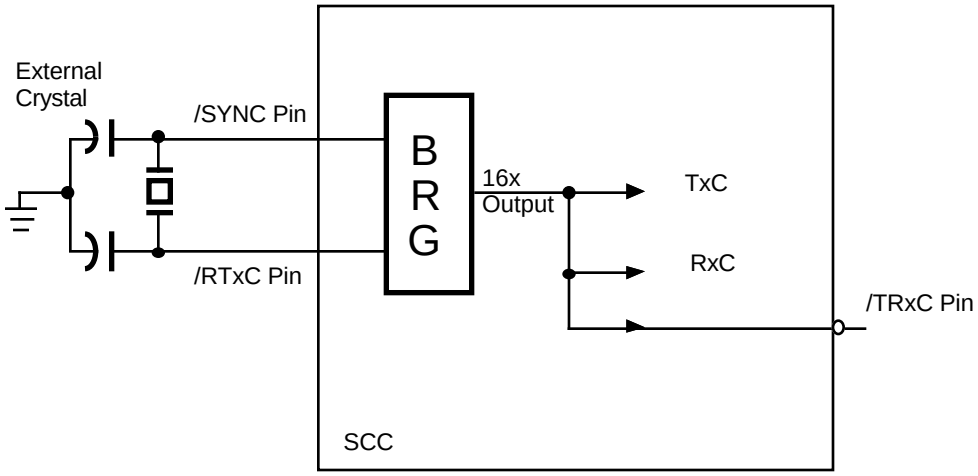



**Figure 3-11. Async Clock Setup Using an External Crystal**

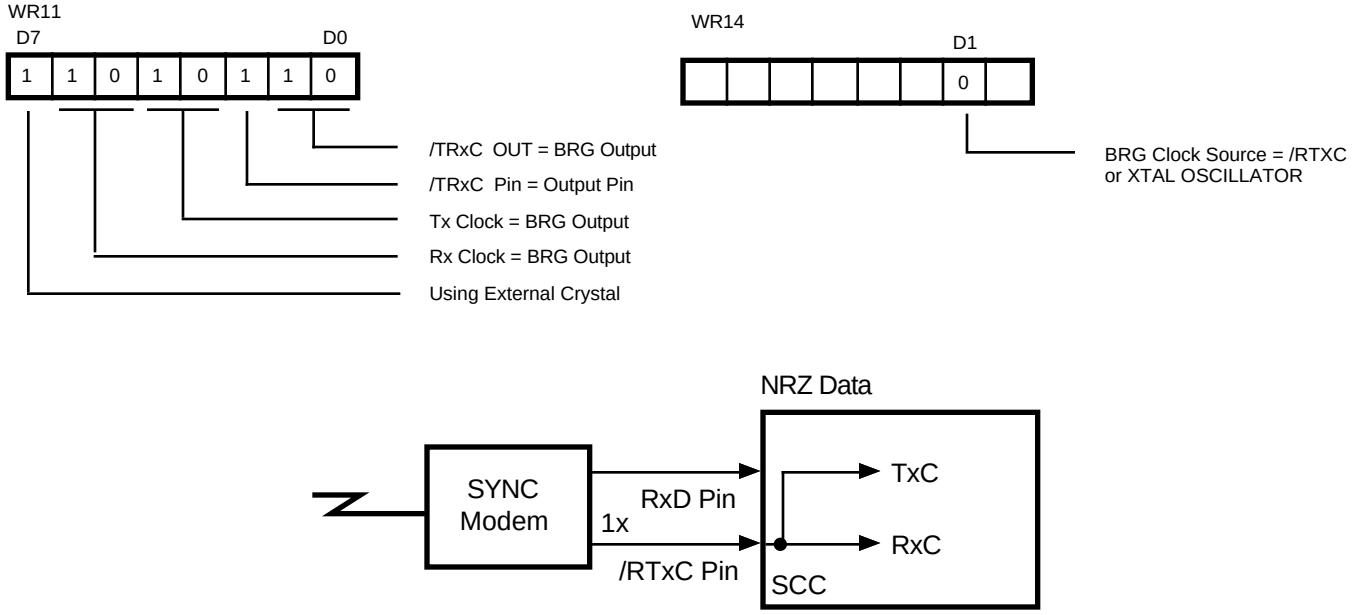



**Figure 3-12. Clock Source Selection**

## 3.6 CRYSTAL OSCILLATOR (Continued)

Figure 3-13 shows the use of the DPLL to derive a 1x clock from the data. In this example:

The DPLL clock input = BRG output (x16 the data rate) WR14.

The DPLL clock output = RxC (receiver clock) WR11.

Set FM mode WR14.

Set FM mode WR10.

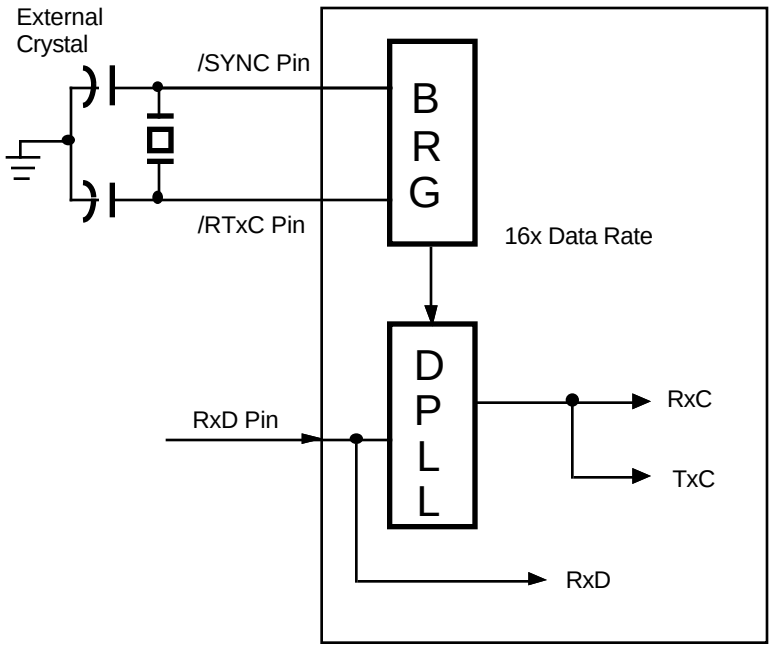



**Figure 3-13. Synchronous Transmission, 1x Clock Rate, FM Data Encoding, using DPLL**

## 3.6 CRYSTAL OSCILLATOR

Each channel contains a high gain oscillator amplifier for use with an external crystal circuit. The amplifier is available between the /RTxC pin (crystal input) and the /SYNC pin (crystal output) for each channel.

The oscillator amplifier is enabled by writing WR11 D7=1. While the crystal oscillator is enabled, anything that has selected the /RTxC pin as its clock source automatically connects to the output of the crystal oscillator.

**Note:** The output of the oscillator amplifier can be programmed to output on the /TRxC pin, which is particularly valuable for diagnostic purposes. Because amplifier characteristics can be affected by the impedance of measurement equipment applied directly to the crystal circuit, using the /TRxC pin allows the oscillation to be tested without affecting the circuit.

Of course, since the oscillator uses the /RTxC and /SYNC pins, this precludes the use of these pins for other functions. In synchronous modes, no sync pulse is output, and the External Sync mode cannot be selected. In asynchronous modes, the state of the Sync/Hunt bit in RR0 is no longer controlled by the /SYNC pin. Instead, the Sync/Hunt bit is forced to 0.

The crystal oscillator requires some finite time to stabilize and must be allowed to stabilize before it is used as a clock source. This stabilization time is dependent on the external circuit impedance and 20 ms is a suggested minimum. The External Crystal should operate in parallel resonance. For further details on designing with the crystal, refer to Appendix A, "On-Chip Oscillator Design".

# CHAPTER 4
## DATA COMMUNICATION MODES

## 4.1 INTRODUCTION

The SCC provides two independent, full-duplex channels programmable for use in any common asynchronous or synchronous data communication protocol. The data communication protocols handled by the SCC are:

- Asynchronous mode:
  Asynchronous (x16, x32, or x64 clock
  Isochronous (x1 clock)
- Character-Oriented mode:
  Monosynchronous
  Bisynchronous
  External Synchronous
- Bit-Oriented mode
  SDLC/HDLC
  SDLC/HDLC Loop

### 4.1.1 Transmit Data Path Description

A diagram of the transmit data path is shown in Figure 4-1. The transmitter has a Transmit Data buffer (a 4-byte deep FIFO on the ESCC, a one byte deep buffer on the NMOS/CMOS version) which is addressed through WR8. It is not necessary to enable the transmit buffer. It is available in all modes of operation. The Transmit Shift register is loaded from either WR6, WR7, or the Transmit Data buffer. In Synchronous modes, WR6 and WR7 are programmed with the sync characters. In Monosync mode, an 8-bit or 6-bit sync character is used (WR6), whereas a 16-bit sync character is used in the Bisynchronous mode (WR6 and WR7). In bit-oriented Synchronous modes, the SDLC flag character (7E hex) is programmed in WR7 and is loaded into the Transmit Shift Register at the beginning and end of each message.

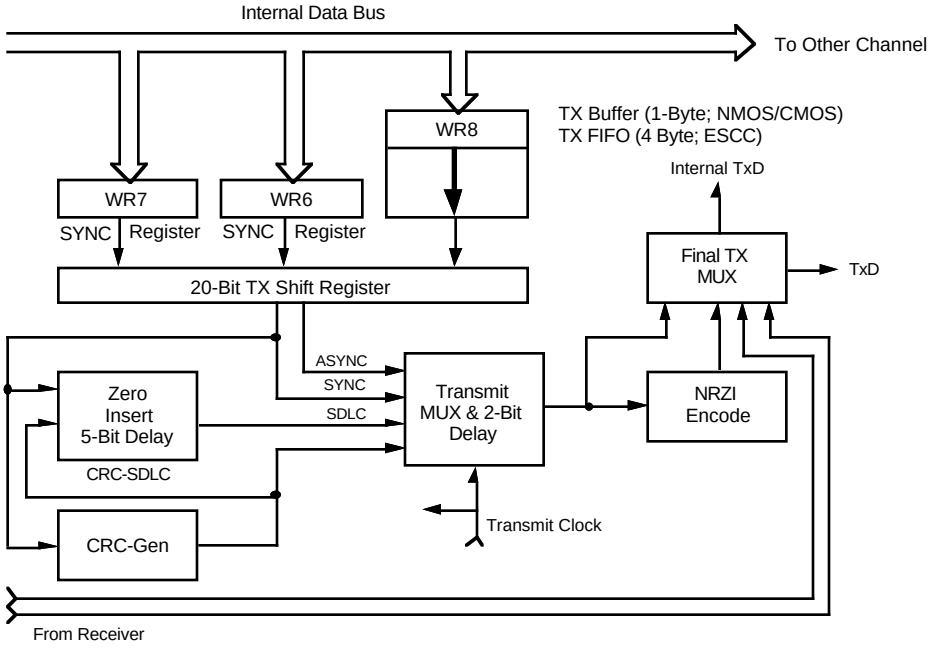



**Figure 4-1. Transmit Data Path**

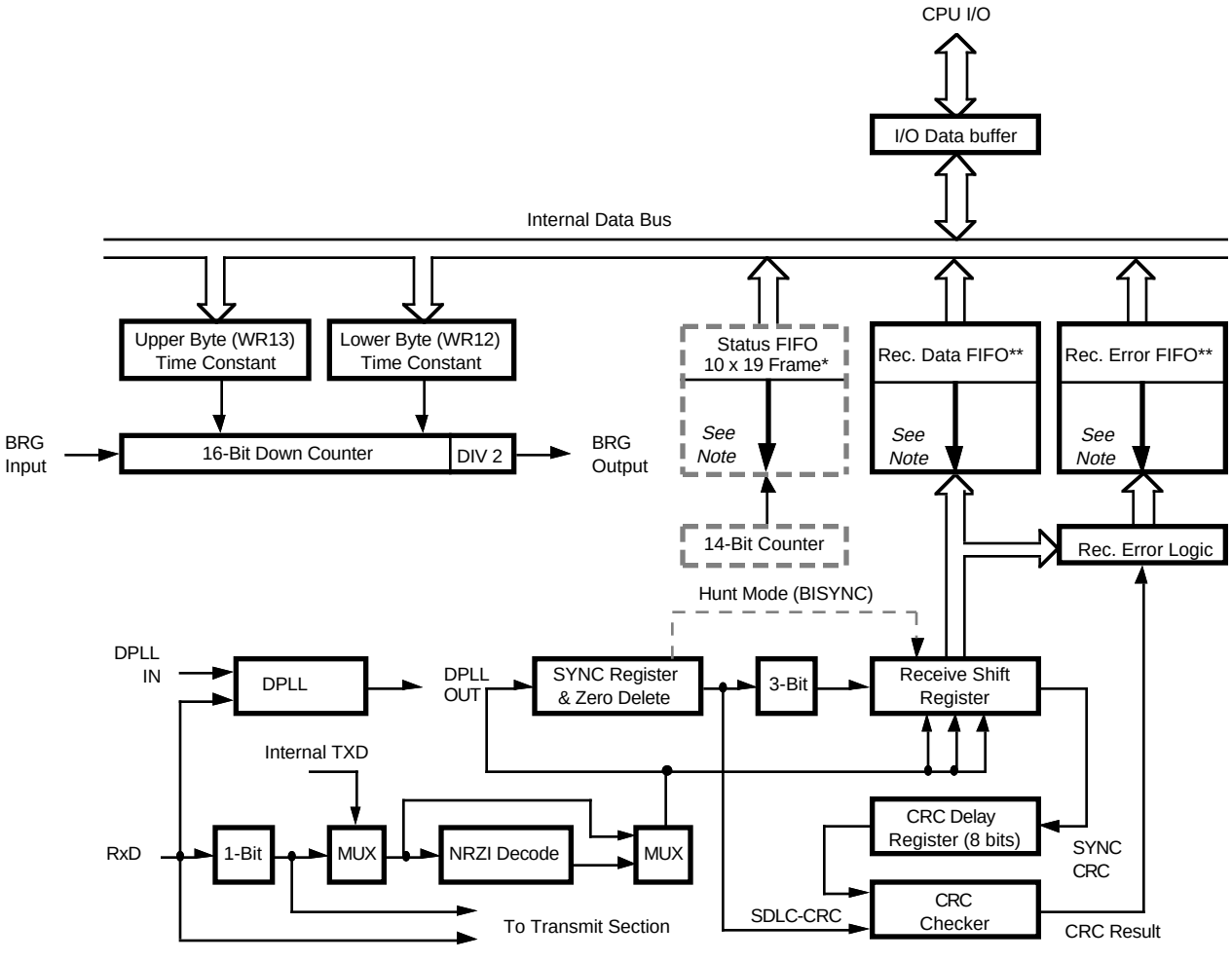
## 4.1 INTRODUCTION (Continued)

For asynchronous data, the Transmit Shift register is formatted with start and stop bits along with the data; optionally with parity information bit. The formatted character is shifted out to the transmit multiplexer at the selected clock rate. WR6 & WR7 are not used in Asynchronous mode.

Synchronous data (except SDLC/HDLC) is shifted to the CRC generator as well as to the transmit multiplexer. SDLC/HDLC data is shifted to the CRC Generator and out through the zero insertion logic (which is disabled while the flags are being sent). A 0 is inserted in all address, control, information, and frame check fields following five contiguous 1s in the data stream. The result of the CRC generator for SDLC data is also routed through the zero insertion logic and then to the transmit multiplexer.

### 4.1.2 Receive Data Path Description

On the ESCC, the receiver has an 8-byte deep, 8-bit wide Data FIFO, while the NMOS/CMOS version receiver has a 3-byte deep, 8-bit wide data buffer. In both cases, the Data buffer is paired with an 8-bit Error FIFO and an 8-bit Shift Register. The receive data path is shown in Figure 4-2. This arrangement creates a 8-character buffer, allowing time for the CPU to service an interrupt or for the DMA to acquire the bus at the beginning of a block of high-speed data. It is not necessary to enable the Receive FIFO, since it is available in all modes of operation. For each data byte in the Receive FIFO, a byte is loaded into the Error FIFO to store parity, framing, and other status information. The Error FIFO is addressed through Read Register 1.




Notes:
* Not with NMOS.
** Rec. Data FIFO and Rec. Error FIFO are 8 Bytes Deep (ESCC), 3 Bytes Deep (NMOS/CMOS).

**Figure 4-2. Receive Data Path**

Incoming data is routed through one of several paths depending on the mode and character length. In Asynchronous mode, serial data enters the 3-bit delay if a character length of seven or eight bits is selected. If a character length of five or six bits is selected, data enters the receive shift register directly.

In Synchronous modes, the data path is determined by the phase of the receive process currently in operation. A synchronous receive operation begins with a hunt phase in which a bit pattern that matches the programmed sync characters (6-,8-, or 16-bit) is searched.

The incoming data then passes through the Sync register and is compared to a sync character stored in WR6 or WR7 (depending on which mode it is in). The Monosync mode matches the sync character programmed in WR7 and the character assembled in the Receive Sync register to establish synchronization.

Synchronization is achieved differently in the Bisync mode. Incoming data is shifted to the Receive Shift register while the next eight bits of the message are assembled in the Receive Sync register. If these two characters match the programmed characters in WR6 and WR7, synchronization is established. Incoming data can then bypass the Receive Sync register and enter the 3-bit delay directly.

The SDLC mode of operation uses the Receive Sync register to monitor the receive data stream and to perform zero deletion when necessary; i.e., when five continuous 1s are received, the sixth bit is inspected and deleted from the data stream if it is 0. The seventh bit is inspected only if the sixth bit equals one. If the seventh bit is 0, a flag sequence has been received and the receiver is synchronized to that flag. If the seventh bit is a 1, an abort or an EOP (End Of Poll) is recognized, depending upon the selection of either the normal SDLC mode or SDLCLoop mode.

**Note:** The insertion and deletion of the zero in the SDLC data stream is transparent to the user, as it is done after the data is written to the Transmit FIFO and before data is read from the Receive FIFO. This feature of the SDLC/HDLC protocol is to prevent the inadvertent sending of an ABORT sequence as part of the data stream. It is also valuable to applications using encoded data to insure a sufficient number of edges on the line to keep a DPLL synchronized on a receive data stream.

The same path is taken by incoming data for both SDLC and SDLC Loop modes. The reformatted data enters the 3-bit delay and is transferred to the Receive Shift register. The SDLC receive operation begins in the hunt phase by attempting to match the assembled character in the Receive Shift Register with the flag pattern in WR7. When the flag character is recognized, subsequent data is routed through the same path, regardless of character length.

Either the CRC-16 or CRC-SDLC (cyclic redundancy check or CRC) polynomial can be used for both Monosync and Bisync modes, but only the CRC-SDLC polynomial is used for SDLC operation. The data path taken for each mode is also different. Bisync protocol is a byte-oriented operation that requires the CPU to decide whether or not a data character is to be included in CRC calculation. An 8-bit delay in all Synchronous modes except SDLC is allowed for this process. In SDLC mode, all bytes are included in the CRC calculation.

## 4.2 ASYNCHRONOUS MODE

In asynchronous communications, data is transferred in the format shown in Figure 4-3.

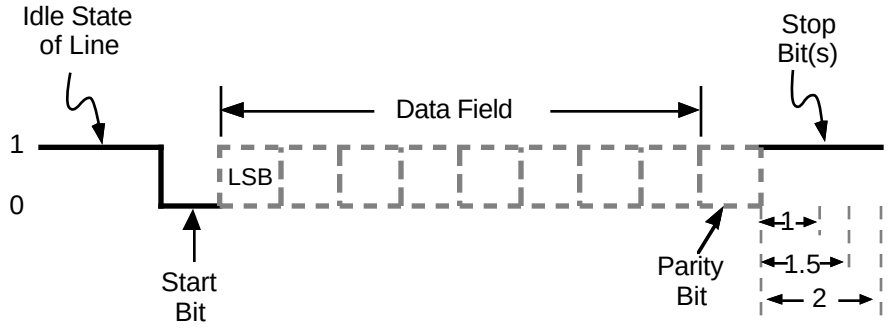



**Figure 4-3. Asynchronous Message Format**

## 4.2 ASYNCHRONOUS MODE (Continued)

The transmission of a character begins when the line makes a transition from the 1 state (or MARK condition) to the 0 state (or SPACE condition). This transition is the reference by which the character's bit cell boundaries are defined. Though the transmitter and receiver have no common clock signal, they must be at the same data rate so that the receiver can sample the data in the center of the bit cell.

The SCC also supports Isochronous mode, which is the same as Asynchronous except that the clock is the same rate as the data. This mode is selected by selecting x1 clock mode in WR4 (D7 & D6=0). Using this mode typically requires that the transmit clock source be transmitted along with the data, or that the clock be synchronized with the data.

The character can be broken up into four fields:

- Start bit - signals the beginning of a character frame.
- Data field - typically 5-8 bits wide.
- Parity bit - optional error checking mechanism.
- Stop bit(s) - Provides a minimum interval between the end of one character and the beginning of the next.

Generation and checking of parity is optional and is controlled by WR4 D1 & D0. WR4 bit D0 is used to enable parity. If WR4 bit D1 is set, even parity is selected and if D1 is reset, odd parity is selected. For even parity, the parity bit is set/reset so that the data byte plus the parity bit contains an even number of 1s. For odd parity, the parity bit is set/reset such that the data byte plus the parity bit contains an odd number of 1s.

The SCC supports Asynchronous mode with a number of programmable options including the number of bits per character, the number of stop bits, the clock factor, modem interface signals, and break detect and generation.

Asynchronous mode is selected by programming the desired number of stop bits in D3 and D2 of WR4. Programming these two bits with other than 00 places both the receiver and transmitter in Asynchronous mode. In this mode, the SCC ignores the state of bits D4, D3, and D2 of WR3, bits D5 and D4 of WR4, bits D2 and D0 of WR5, all of WR6 and WR7, and all of WR10 except D6 and D5. Ignored bits are programmed with 1 or 0 (Table 4-1).

**Table 4-1. Write Register Bits Ignored in Asynchronous Mode**

| Register | D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 |
|---|---|---|---|---|---|---|---|---|
| WR3 | | | | x | x | x | 0 | |
| WR4 | | | x | x | | | | |
| WR5 | | | | | | x | | x |
| WR6 | x | x | x | x | x | x | x | x |
| WR7 | x | x | x | x | x | x | x | x |
| WR10 | x | | | x | x | x | x | x |

**Note:** If WR3 D1 is set (enabling the sync character load inhibit feature), any character matching the value in WR6 is stripped out of the incoming data stream and not put into the Receive FIFO. Therefore, as this feature is typically only desired in synchronous formats, this bit should reset in Asynchronous mode.

### 4.2.1 Asynchronous Transmit

Asynchronous mode is selected by specifying the number of stop bits per character in bits D3 and D2 of WR4. The three options available are one, one-and-a-half, and two stop bits per character. These two bits select only the number of stop bits for the transmitter, as the receiver always checks for one stop bit.

The number of bits per transmitted character is controlled both by bits D6 and D5 in WR5 and the way the data is formatted within the transmit buffer (in the case of the ESCC, Transmit FIFO). The bits in WR5 allow the option of five, six, seven, or eight bits per character. In all cases the data must be right-justified, with the unused bits being ignored except in the case of five bits per character. When the five bits per character option is selected, the data may be formatted before being written to the transmit buffer. This allows transmission of from one to five bits per character. The formatting is shown in Table 4-2.

**Table 4-2. Transmit Bits per Character**

| Bit 7 | Bit 6 | |
|---|---|---|
| 0 | 0 | 5 or less bits/character |
| 0 | 1 | 7 bits/character |
| 1 | 0 | 6 bits/character |
| 1 | 1 | 8 bits/character |

**Note:** For five or less bits per character selection in WR5, the following encoding is used in the data sent to the transmitter. D is the data bit(s) to be sent.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | D | Sends one data bit |
| 1 | 1 | 1 | 0 | 0 | 0 | D | D | Sends two data bits |
| 1 | 1 | 0 | 0 | 0 | D | D | D | Sends three data bits |
| 1 | 0 | 0 | 0 | D | D | D | D | Sends four data bits |
| 0 | 0 | 0 | D | D | D | D | D | Sends five data bits |

An additional bit, carrying parity information, may be automatically appended to every transmitted character by setting bit D0 of WR4 to 1. This bit is sent in addition to the number of bits specified in WR4 or by bit D1 of WR4. If this bit is set to 1, the transmitter sends even parity and, if set to 0, the parity is odd.

The transmitter may be programmed to send a Break by setting bit D4 of WR5 to 1. The transmitter will send contiguous 0s from the first transmit clock edge after this command is issued, until the first transmit clock edge after this bit is reset. The transmit clock edges referred to here are those that defined transmitted bit cell boundaries. Care must be taken when Break is sent. As mentioned above, the SCC initiates the Break sequence regardless of the character boundaries. Typically, the break sequence is defined as "null character (all 0 data) with framing error". The other party may not be able to recognize it as a break sequence if the Send Break bit has been set in the middle of sending a non-zero character.

An additional status bit for use in Asynchronous mode is available in bit D0 of RR1. This bit, called All Sent, is set when the transmitter is completely empty and any previous data or stop bits have reached the TxD pin. The All Sent bit can be used by the processor as an indication that the transmitter may be safely disabled, or indication to change the modem status signal.

The SCC may be programmed to accept a transmit clock that is one, sixteen, thirty-two, or sixty-four times the data rate. This is selected by bits D7 and D6 in WR4, in common with the clock factor for the receiver.

**Note:** When using Isosynchronous (X1 clock) mode, one-and-a-half stop bits are not allowed. Only one or two stop bits should be selected. If some length other than one stop bit is desired in the times one mode, only two stop bits may be used. Also, in this mode, the Transmitter usually needs to send clocking information (transmit clock) along with the data in order to receive data correctly.

There are two modem control signals associated with the transmitter provided by the SCC; /RTS and /CTS.

The /RTS pin is a simple output that carries the inverted state of the RTS bit (D1) in WR5, unless the Auto Enables mode bit (D5) is set in WR3. When Auto Enables is set, the /RTS pin immediately goes Low when the RTS bit is set. However, when the RTS bit is reset, the /RTS pin remains Low until the transmitter is completely empty and the last stop bit has left the TxD pin. Thus, the /RTS pin may be used to disable external drivers for the transmit data. The /CTS pin is ordinarily a simple input to the CTS bit in RR0. However, if Auto Enables mode is selected, this pin becomes an enable for the transmitter. That is, if Auto Enables is on and the /CTS pin is High, the transmitter is disabled; the transmitter is enabled while the /CTS pin is Low.

The initialization sequence for the transmitter in Asynchronous mode is WR4 first to select the mode, then WR3 and WR5 to select the various options. At this point the other registers should be initialized as necessary. When all of this is complete, the transmitter may be enabled by setting bit D3 of WR5 to 1. Note that the transmitter and receiver may be initialized at the same time.

#### 4.2.1.1 Asynchronous transmit on the NMOS/CMOS

On the NMOS/CMOS version of the SCC, characters are loaded from the transmit buffer to the shift register where they are given a start bit and a parity bit (as programmed), and are shifted out to the TxD pin. The transmit buffer empty interrupt and the DMA request (either /W//REQ or /DTR//REQ pin) are asserted when the transmit buffer is empty, if these are enabled. At this time, the CPU or the DMA is able to write one byte of transmit data. The Transmit Buffer Empty (TBE) bit (RR0, bit D2) also follows the state of the transmit buffer. The All Sent bit, RR1, bit D0, can be polled to determine when the last bit of transmit data has cleared the TxD pin. For details about the transmit DMA and transmit interrupts, refer to Section 2.4.8 "Transmit Interrupt and Transmit Buffer Empty bit."

#### 4.2.1.2 Asynchronous transmit on the ESCC

On the ESCC, characters are loaded from the Transmit FIFO to the shift register where they are given a start bit and a parity bit (as programmed), and are shifted out to the TxD pin. The ESCC can generate an interrupt or DMA request depending on the status of the Transmit FIFO. If WR7' D5 is reset, the transmit buffer empty interrupt and DMA request (either /W//REQ or /DTR//REQ pin) are asserted when the entry location of the Transmit FIFO is empty (one byte can be written). If WR7' D5 is set, the transmit interrupt and DMA request is generated when the Transmit FIFO is completely empty (four bytes can be written). The Transmit Buffer Empty (TBE) bit in RR0, bit D2 also is affected by the state of WR7' bit D5. The All Sent

## 4.2 ASYNCHRONOUS MODE (Continued)

bit, bit D0 of RR1, can be polled to determine when the last bit of transmit data has cleared the TxD pin.

The number of transmit interrupts can be minimized by setting bit D5 of WR7' to one and writing four bytes to the transmitter for each transmit interrupt. This requires that the system response to interrupt is less than the time it takes to transmit one byte at the programmed baud rate. If the system's interrupt response time is too long to use this feature, bit D5 of WR7' should be reset to 0. Then, poll the TBE bit and poll after each data write to test if there is space in the Transmit FIFO for more data.

For details about the transmit DMA and transmit interrupts, refer to Section 2.4.8 "Transmit Interrupt and Transmit Buffer Empty bit".

### 4.2.2 Asynchronous Receive

Asynchronous mode is selected by specifying the number of stop bits per character in bits D3 and D2 of WR4. This selection applies only to the transmitter, however, as the receiver always checks for one stop bit. If after character assembly the receiver finds this stop bit to be a 0, the Framing Error bit in the receive error FIFO is set at the same time that the character is transferred to the receive data FIFO. This error bit accompanies the data to the exit location (CPU side) of the Receive FIFO, where it is a special receive condition. The Framing Error bit is not latched, so it must be read in RR1 before the accompanying data is read.

The number of bits per character is controlled by bits D7 and D6 of WR3. Five, six, seven or eight bits per character may be selected via these two bits. Data is right justified with the unused bits set to 1s. An additional bit, carrying parity information, may be selected by setting bit D0 of WR4 to 1. Note that this also enables parity for the transmitter. The parity sense is selected by bit D1 of WR4. If this bit is set to 1, the received character is checked for even parity, and if set to 0, the received character is checked for odd parity. The additional bit per character that is parity is transferred to the receive data FIFO along with the data, if the data plus parity is eight bits or less. The parity error bit in the receive error FIFO may be programmed to cause special receive interrupts by setting bit D2 of WR1 to 1. Once set, this error bit is latched and remains active until an Error Reset command has been issued.

Since errors apply to specific characters, it is necessary that error information moves alongside the data that it refers to. This is implemented in the SCC with an error FIFO in parallel with the data FIFO. The three error conditions that the receiver checks for in Asynchronous mode are:

- Framing errors—When a character's stop bit is a 0.
- Parity errors—The parity bit of a character disagrees with the sense programmed in WR4.
- Overrun errors—When the Receive FIFO overflows.

If interrupts are not used to transfer data, the Parity Error, Framing Error, and Overrun Error bits in RR1 should be checked before the data is removed from the receive data FIFO, because reading data pops up the error information stored in the Error FIFO.

The SCC may be programmed to accept a receive clock that is one, sixteen, thirty-two, or sixty-four times the data rate. This is selected by bits D7 and D6 in WR4. The 1X mode is used when bit synchronization external to the received clock is present (i.e., the clock recovery circuit, or active receive clock from the sender side). The 1X mode is the only mode in which a data encoding method other than NRZ may be used. The clock factor is common to the receiver and transmitter.

The break condition is continuous 0s, as opposed to the usual continuous ones during an idle condition. The SCC recognizes the Break condition upon seeing a null character (all 0s) plus a framing error. Upon recognizing this sequence, the Break bit in RR0 is set and remains set until a 1 is received. At this point, the break condition is no longer present. At the termination of a break, the receive data FIFO contains a single null character, which should be read and discarded. The framing error bit will not be set for this character, but if odd parity has been selected, the Parity Error bit is set.

**Note:** Caution should be exercised if the receive data line contains a switch that is not debounced to generate breaks. If this is the case, switch bounce may cause multiple breaks to be recognized by the SCC, with additional characters assembled in the receive data FIFO and the possibility of a receive overrun condition being latched.

The SCC provides up to three modem control signals associated with the receiver; /SYNC, /DTR//REQ, and /DCD.

The /SYNC pin is a general purpose input whose state is reported in the Sync/Hunt bit in RR0. If the crystal oscillator is enabled, this pin is not available and the Sync/Hunt bit is forced to 0. Otherwise, the /SYNC pin may be used to carry the Ring Indicator signal.

The /DTR//REQ pin carries the inverted state of the DTR bit (D7) in WR5 unless this pin has been programmed to carry a DMA request signal.

The /DCD pin is ordinarily a simple input to the DCD bit in RR0. However, if the Auto Enables mode is selected by setting D5 of WR3 to 1, this pin becomes an enable for the

receiver. That is, if Auto Enables is on and the /DCD pin is High, the receiver is disabled; while the /DCD pin is low, the receiver is enabled.

Received characters are assembled, checked for errors, and moved to the receive data FIFO (eight bytes on ESCC, three bytes on NMOS/CMOS). The user can program the SCC to generate an interrupt to the CPU or to request a data read from a DMA when data is received.

On the NMOS/CMOS version, it generates the Receive Character Available interrupt and DMA Request on Receive (if enabled). The receive interrupt and DMA request is generated when there is at least one character in the FIFO. The Rx Character Available (RCA) bit is set if there is at least one byte available.

The ESCC generates the receive character available interrupt and DMA request on Receive (if enabled) and is dependent on WR7' bit D3. If this bit is reset to 0 (this mode is comparable to the NMOS/CMOS version), the receive interrupt and DMA request is generated when there is at least one character in the FIFO. If WR7' bit D3 is set to 1, the receive interrupt and DMA request are generated when there are four bytes available in the Receive FIFO. The RCA bit in RR0 follows the state of WR7' D3. The RCA bit is set if there is at least one byte available, regardless of the status of WR7' bit D3.

This is the initialization sequence for the receiver in Asynchronous mode. First, WR4 selects the mode, then WR3 and WR5 select the various options. At this point, the other registers should be initialized as necessary. When all of this is complete, the receiver may be enabled by setting bit D0 of WR3 to 1.

See Section 2.4.7 “The Receive Interrupt” for more details on receive interrupts.

### 4.2.3 Asynchronous Initialization

The initialization sequence for Asynchronous mode is shown in Table 4-3. All of the SCC's registers should be re-initialized after a channel or hardware reset. Also, WR4 should be programmed first after a reset.

**Table 4-3. Initialization Sequence Asynchronous Mode**

| Reg | Bit No | Description |
|---|---|---|
| WR9 | 6, 7 | Hardware or channel Reset |
| WR4 | 3, 2 | Select Async Mode and the number of stop bits* |
| | 0, 1 | Select parity* |
| | 6, 7 | Select clock mode* |
| WR3 | 7, 6 | Select number of receive bits per character |
| | 5 | Select Auto Enables Mode* |
| WR5 | 6, 5 | Select number of bits/char for transmitter |
| | 1 | Select modem control (RTS) |

**Note:**
* Initializes transmitter and receiver simultaneously.

At this point, the other registers should be initialized according to the hardware design such as clocking,
I/O mode, etc. When this is completed, the transmitter is enabled by setting WR5 bit D3 to 1 and the receiver is enabled by setting WR3 bit D0 to 1.

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE

The SCC supports three byte-oriented synchronous protocols. They are: monosynchronous, bisynchronous, and external synchronous.

In synchronous communications, the bit cell boundaries are referenced to a clock signal common to both the transmitter and receiver. Consequently, they operate in a fixed-phase relationship. This eliminates the need for the receiver to locate the bit cell boundaries with a clock 16, 32, or 64 times the receive data rate, allowing for higher speed communication links. Some applications may encode (i.e., NRZI or FM coding) the clock information on the same line as the data. Therefore, these applications require that the receiver use a high speed clock to find the bit cell boundaries (decoding is typically done with the PLL—Phase-Locked Loop; the SCC has on-chip Digital PLL). Data encoding eliminates the need to transmit the synchronous clock on a separate wire from the data.

Synchronous data does not use start and stop bits to delineate the boundaries for each character. This eliminates the overhead associated with every character and increases the line efficiency. Because of the phase relationship of synchronous data to a clock, data is transferred in blocks with no gaps between characters. This requires that there be an agreement as to the location of the character boundaries so that the characters can be properly framed. This is normally accomplished by defining special synchronization patterns, or Sync characters. The synchronization pattern serves as a reference; it signals the receiver that a character boundary occurs immediately after the last bit of the pattern. For example Monosync Protocol usually uses 16 Hex as this special character, and the SDLC protocol uses 0, six 1s, followed by a 0 (7E Hex; usually referred to as Flag Pattern) to mark the beginning and end of a block of data. Another way of identifying the character boundaries (i.e., achieving synchronization) is with a logic signal that goes active just as the first character is about to enter the receiver. This method is referred to as External Synchronization.

Figure 4-4 shows the character format for synchronous transmission. For example, bits 1-8 might be one character and bits 9-13 part of another character; or, bit 1 might be part of a second character, and bits 10-13 part of a third character. This is accomplished by defining a synchronization character, commonly called a Sync Character.

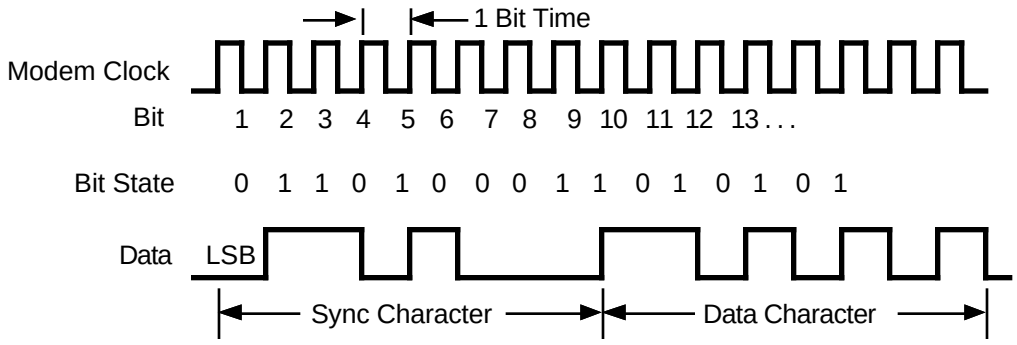



**Figure 4-4. Monosync Data Character Format**

### 4.3.1 Byte-Oriented Synchronous Transmit

Once Synchronous mode has been selected, any of three of the following sync character lengths may be selected:

- 6-bit
- 8-bit
- 16-bit

The 6-bit option sync character is selected by setting bits 4 and 5 of WR4 to zeros and bit 0 of WR10 to one. Only the least significant six bits of WR6 are transmitted.

The 8-bit sync character is selected by setting bits 4 and 5 of WR4 to zeros and bit 0 of WR10 to zeros. With this option selected, the transmitter sends the contents of WR6 when it has no data to send.

For a 16-bit sync character, set bit D4 of WR4 to 1 and bit D5 of WR4 and bit D0 of WR10 to 0. In this mode, the transmitter sends the concatenation of WR6 and WR7 for the idle line condition.

Because the receiver requires that sync characters be left-justified in the registers, while the transmitter requires them to be right justified, only the receiver works with a 12-bit sync character. While the receiver is in External Sync

mode, the transmitter sync length may be six or eight bits, as selected by bit D0 of WR10.

Monosync and Bisync modes require clocking information to be transmitted along with the data either by a method of encoding data that contains clocking information, or by a modem that encodes or decodes clock information in the modulation process. Refer to the Monosync message format shown in Figure 4-4.

The Bisync mode of operation is similar to the Monosync mode, except that two sync characters are provided instead of one. Bisync attempts a more structured approach to synchronization through the use of special characters as message headers or trailers.

Character-oriented mode is selected by programming bits D3 and D2 of WR4 with zeros. This selects Synchronous mode, as opposed to Asynchronous mode, but this selection is further modified by bits 5 and 7 of WR4 as well as bits 1 and 0 of WR10. During the sync character-oriented modes, except in External Sync mode, the state of bits 7 and 6 of WR4 are always forced internally to zeros. In external sync mode, these two bits must be programmed with zeros (Table 4-4.). The combination, other than 00 in External Sync mode, puts the SCC in special synchronization modes.

**Table 4-4. Registers Used in Character-Oriented Modes**

| Reg | Bit No | Description |
|---|---|---|
| WR4 | 3 (=0) | select sync mode |
| | 2 (=0) | |
| | 4 (=0) | select monosync mode |
| | 5 (=0) | (8-bit sync character) |
| | 4 (=1) | select bisync mode |
| | 5 (=0) | (16-bit sync character) |
| | 4 (=1) | select external sync mode |
| | 5 (=1) | (external sync signal required) |
| | 6 (=0) | select 1x clock mode |
| | 7 (=0) | |
| WR6 | 7-0 | sync character (low byte) |
| WR7 | 7-0 | sync character (high byte) |
| WR10 | 1 | select sync character length |

In character-oriented modes, a special bit pattern is used to provide character synchronization. The SCC offers several options to support Synchronous mode including various sync generation and checking, CRC generation and checking, as well as modem controls and a transmitter to receiver synchronization function.

The number of bits per transmitted character is controlled by D6 and D5 of WR5 plus the way the data is formatted within the transmit buffer. The bits in WR5 select the option of five, six, seven, or eight bits per character. In all cases, the data must be right-justified, with the unused bits being ignored except in the case of five bits per character. When the five bits per character option is selected, the data must be formatted before being written to the transmit buffer to allow transmission of from one to five bits per character. This formatting is shown in Table 4-2.

An additional bit, carrying parity information, may be automatically appended to every transmitted character by setting bit D0 of WR4 to 1. This parity bit is sent in addition to the number of bits specified in WR4 or by the data format. If this bit is set to 1, the transmitter sends even parity; if set to 0, the transmitted parity is odd. Parity is not typically used in synchronous applications because the CRC provides a more reliable method for detecting errors.

Either of two CRC polynomials are used in Synchronous modes, selected by bit D2 in WR5. If this bit is set to 1, the CRC-16 polynomial is used and, if this bit is set to 0, the CRC-CCITT polynomial is used. This bit controls the selection for both the transmitter and receiver. The initial state of the generator and checker is controlled by bit D7 of WR10. When this bit is set to 1, both the generator and checker have an initial value of all ones; if this bit is set to 0, the initial values are all zeros.

The SCC does not automatically preset the CRC generator in byte Synchronous modes, so this must be done in software. This is accomplished by issuing the Reset Tx CRC Generator command, which is encoded in bits D7 and D6 of WR0. For proper results, this command is issued while the transmitter is enabled and sending sync characters.

If the CRC is to be used, the transmit CRC generator must be enabled by setting bit D0 of WR5 to 1. This bit may also be used to exclude certain characters from the CRC calculation. Sync characters (from sync registers) are automatically excluded from the CRC calculation, and any characters written as data are excluded from the calculation by using bit D0 of WR5. Internally, enabling or disabling the CRC for a particular character happens at the same time the character is loaded from the transmit data buffer (on the ESCC, the Transmit FIFO) to the Transmit Shift register. Thus, to exclude a character from the CRC calculation bit, D0 of WR5 is set to 0 before the character is written to the transmit buffer (on the ESCC, the Transmit FIFO).

***ESCC:***
***Since the ESCC has a four-byte FIFO, if a character is to be excluded from the CRC calculation, it is recommended that only one byte be written to the ESCC at that time. If WR7' D5 is reset, the transmit interrupt is generated when the FIFO is completely empty. This can be used as a signal to reset WR5 bit D0, and then the character can be written to the Transmit FIFO. This guarantees that the internal disable occurs when the character moves from the buffer to the shift register.***

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

***Once the buffer becomes empty, the Tx CRC Enable bit is written for the next character.***

Enabling the CRC generator is not sufficient to control the transmission of the CRC. In the SCC, this function is controlled by the Tx Underrun/EOM bit, which is reset by the processor and set by the SCC. When the transmitter underruns (both the transmit buffer and Transmit Shift register are empty) the state of the Tx Underrun/EOM bit determines the action taken by the SCC. If the Tx Underrun/EOM bit is reset when the underrun occurs, the transmitter sends the accumulated CRC and sets the Tx Underrun/EOM bit to indicate this. This transition is programmed to cause an external/status interrupt, or the Tx Underrun/EOM is available in RR0.

The Reset Tx Underrun/EOM Latch command is encoded in bits D7 and D6 of WR0. For correct transmission of the CRC at the end of a block of data, this command is issued after the first character is written to the SCC but before the transmitter underruns. The command is usually issued immediately after the first character is written to the SCC so that the CRC is sent if an underrun occurs inadvertently during the block of data.

***85X30***
***If WR7' bit D1 is set, the Reset Transmit Underrun/EOM latch is automatically reset after the first byte is written to the transmitter. This eliminates the need for the CPU to issue this command. This feature can be particularly useful to applications using a DMA to write data to the transmitter since there is no longer a need to interrupt the data transfers to issue this command.***

If the transmitter is disabled during the transmission of a character, that character is sent completely. This applies to both data and sync characters. However, if the transmitter is disabled during the transmission of the CRC, the 16-bit transmission is completed, but the remaining bits will come from the Sync registers rather than the remainder of the CRC.

There are two modem control signals associated with the transmitter provided by the SCC: /RTS and /CTS.

The /RTS pin is a simple output that carries the inverted state of the RTS bit (D1) in WR5.

The /CTS pin is ordinarily a simple input to the CTS bit in RR0. However, if Auto Enables mode is selected, this pin becomes an enable for the transmitter. That is, if Auto Enables is on and the /CTS pin is High, the transmitter is disabled. While the /CTS pin is Low, the transmitter is enabled.

The initialization sequence for the transmitter in character-oriented mode is shown in Table 4-5.

**Table 4-5. Transmitter Initialization in Character-Oriented Mode**

| Reg | Bit No | Description |
|---|---|---|
| WR4 | 0,1 | selects parity (not typically used insync modes) |
| WR5 | 1 | RTS |
| | 2 | selects CRC generator |
| | 5,6 | selects number of bits per character |
| WR10 | 7 | CRC preset value |

At this point, the other registers should be initialized as necessary. When all of this is completed, the transmitter is enabled by setting bit 3 of WR5 to one. Now that the transmitter is enabled, the CRC generator is initialized by issuing the Reset Tx CRC Generator command in WR0, bits 6-7.

### 4.3.2 Byte-Oriented Synchronous Receive

The receiver in the SCC searches for character synchronization only while it is in Hunt mode. In this mode the receiver is idle except that it is searching the incoming data stream for a sync character match.

In Hunt mode, the receiver shifts for each bit into the Receive Shift register. The contents of the Receive Shift register are compared with the sync character (stored in another register), repeating the process until a match occurs. When a match occurs, the receiver begins transferring bytes to the Receive FIFO.

The receiver is in Hunt mode when it is first enabled, and it may be placed in Hunt mode by the processor issuing the Enter Hunt Mode command in WR3. This bit (D4) is a command, so writing a 0 to it has no effect. The hunt status of the receiver is reported by the Sync/Hunt bit in RR0. Sync/Hunt is one of the possible sources of external/status interrupts, with both transitions causing an interrupt. This is true even if the Sync/Hunt bit is set as a result of the processor issuing the Enter Hunt Mode command.

Once the sync character-oriented mode has been selected, any of the four sync character lengths may be selected: 6 bits, 8 bits, 12 bits, or 16 bits.

The Table 4-6 shows the write register bit setting for selecting sync character length.

**Table 4-6. Sync Character Length Selection**

| Sync Length | WR4,D5 | WR4,D4 | WR10,D0 |
|---|---|---|---|
| 6 bits | 0 | 0 | 1 |
| 8 bits | 0 | 0 | 0 |
| 12 bits | 0 | 1 | 1 |
| 16 bits | 0 | 1 | 0 |

The arrangement of the sync character in WR6 and WR7 is shown in Figure 4-5.

For those applications requiring any other sync character length, the SCC makes provision for an external circuit to provide a character synchronization signal on the /SYNC pin. This mode is selected by setting bits D5 and D4 of WR4 to 1. In this mode, the Sync/Hunt bit in RR0 reports the state of the /SYNC pin, but the receiver is still placed in Hunt mode when the external logic is searching for a sync character match. Two receive clock cycles after the last bit of the sync character is received, the receiver is in Hunt mode and the /SYNC pin is driven Low, then character assembly begins on the rising edge of the receive clock. This immediately precedes the activation of /SYNC (Figure 4-6). The receiver leaves Hunt mode when /SYNC is driven Low.

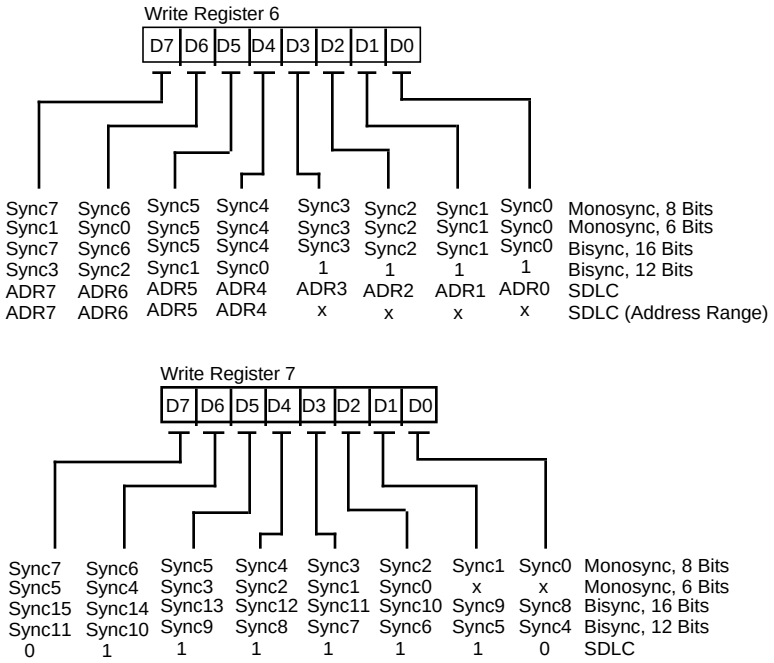



**Figure 4-5. Sync Character Programming**

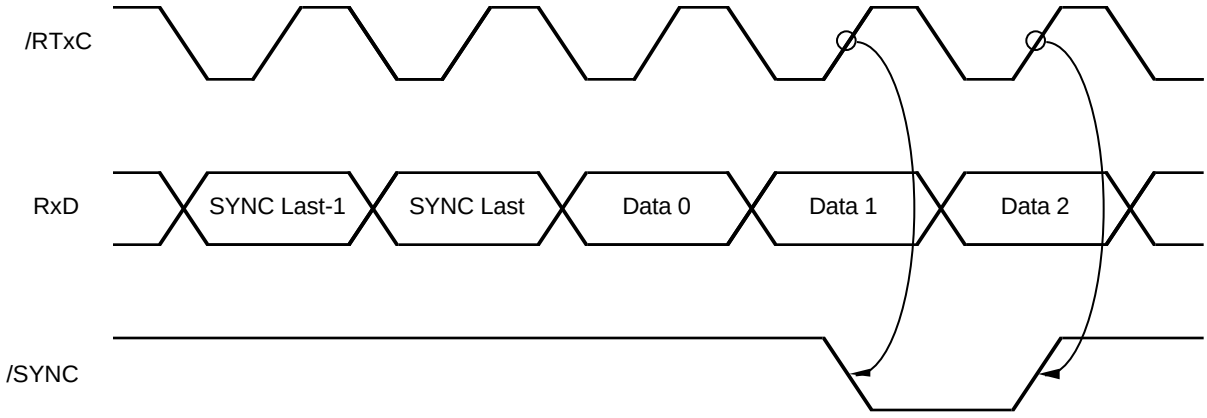



**Figure 4-6. /SYNC as an Input**

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

In all cases except External Sync mode, the /SYNC pin is an output that is driven Low by the SCC to signal that a sync character has been received. The /SYNC pin is activated regardless of character boundaries, so any external circuitry using it should only respond to the /SYNC pulse that occurs while the receiver is in Hunt mode. The timing for the /SYNC signal is shown in Figure 4-7.

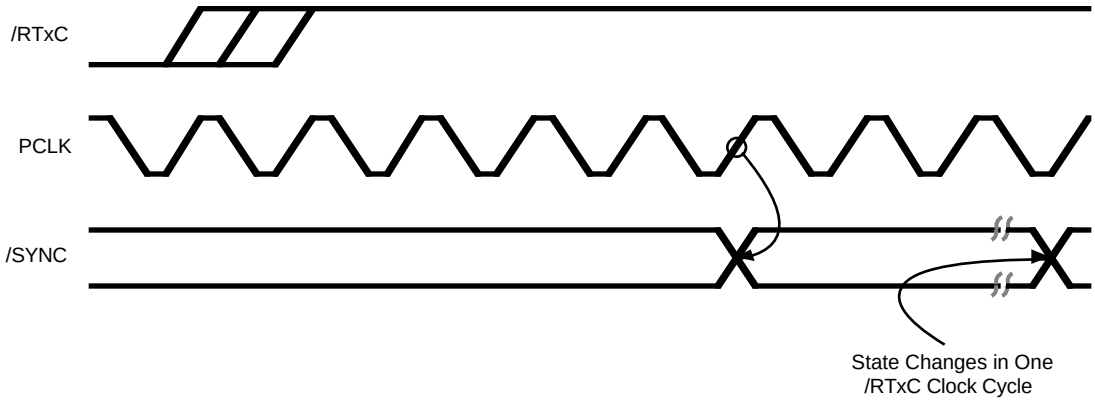



**Figure 4-7. /SYNC as an Output**

To prevent sync characters from entering the receive data FIFO, set the Sync Character Load Inhibit bit (D1) in WR3 to 1. While this bit is set to 1, characters about to be loaded into the receive data FIFO are compared with the contents of WR6. If all eight bits match the character, it is not loaded into the receive data FIFO. Because the comparison is across eight bits, this function should only be used with 8-bit sync characters. It cannot be used with 12- or 16-bit sync characters. Both leading sync characters are removed in the case of a 6-bit sync character. Care must be exercised in using this feature because sync characters which are not transferred to the receive data FIFO will automatically be excluded from CRC calculation. This works properly only in the 8-bit case.

The number of bits per character is controlled by bits D7 and D6 of WR3. Five, six, seven, or eight bits per character may be selected via these two bits. The data is right-justified in the receive data buffer. The SCC merely takes a snapshot of the receive data stream at the appropriate times, so the "unused" bits in the receive buffer are only the bits following the character in the data stream.

An additional bit carrying parity information is selected by setting bit D0 of WR4 to 1. Note that this also enables parity for the transmitter. The bit D1 of WR4 selects parity sense. If this bit is set to 1, the received character is checked for even parity. If WR4 D1 is reset to 0, the received character is checked for odd parity. The additional bit per character is transferred to the FIFO as a part of data when the data plus parity is less than 8 bits per character. The Parity Error bit in the receive error FIFO may be programmed to cause a Special Receive Condition interrupt by setting bit D2 of WR1 to 1. Once set, this error bit is latched and remains active until an Error Reset command has been issued. If interrupts are not used to transfer data, the Parity Error, CRC Error, and Overrun Error bits in RR1 should be checked before the data is removed from the receive data FIFO.

The character length can be changed at any time before the new number of bits has been assembled by the receiver, but, care should be exercised as unexpected results may occur. A representative example would be switching from five bits to eight bits and back to five bits (Figure 4-8).

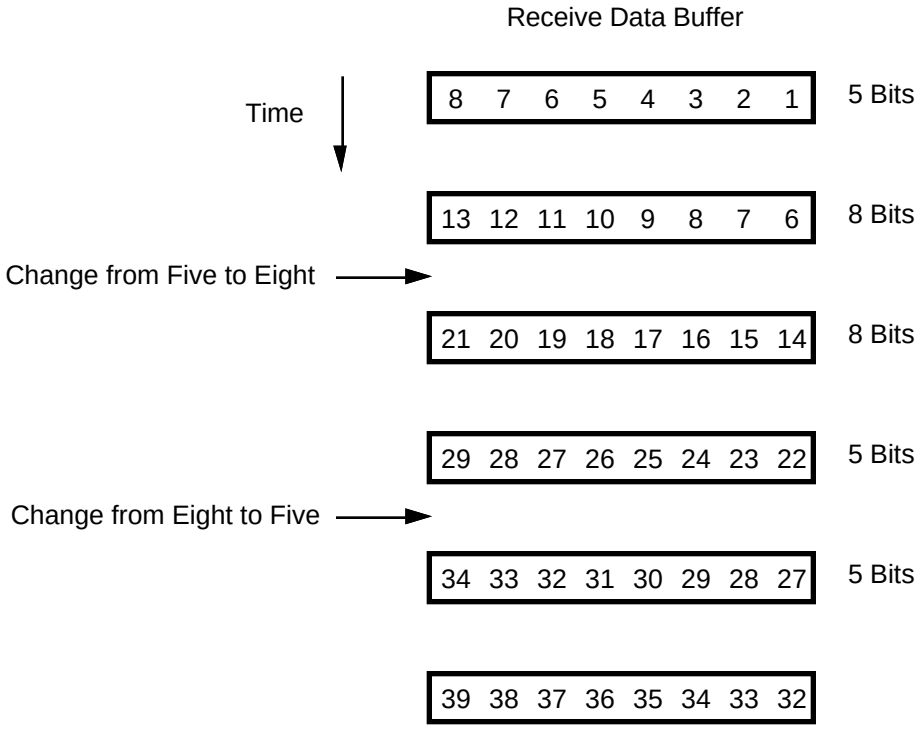



**Figure 4-8. Changing Character Length**

Either of two CRC polynomials are used in Synchronous modes, selected by bit D2 in WR5. If this bit is set to 1, the CRC-16 polynomial is used, if this bit is set to 0, the CRC-CCITT polynomial is used. This bit controls the polynomial selection for both the receiver and transmitter.

The initial state of the generator and checker is controlled by bit D7 of WR10. When this bit is set to 1, both the generator and checker have initial values of all ones; if this bit is set to 0, the initial values are all 0. The SCC presets the checker whenever the receiver is in Hunt mode so a CRC reset command is not necessary. However, there is a Reset CRC Checker command in WR0. This command is encoded in bits D7 and D6 of WR0. If the CRC is used, the CRC checker is enabled by setting bit D0 of WR3 to 1.

Sync characters can be stripped from the data stream any time before the first non-sync character is received. If the sync strip feature is not being used, the CRC is not enabled until after the first data character has been transferred to the receive data FIFO. As previously mentioned, 8-bit sync characters stripped from the data stream are automatically excluded from CRC calculation.

Some synchronous protocols require that certain characters be excluded from CRC calculation. This is possible in the SCC because CRC calculations are enabled and disabled on the fly. To give the processor sufficient time to decide whether or not a particular character should be included in the CRC calculation, the SCC contains an 8-bit time delay between the receive shift register and the CRC checker. The logic also guarantees that the calculation only starts or stops on a character boundary by delaying the enable or disable until the next character is loaded into the receive data FIFO. Because the nature of the protocol requires that CRC calculation disable/enable be selected before the next character gets loaded into the Receive FIFO, users cannot take advantage of the FIFO.

To understand how this works refer to Figure 4-9 and the following explanation. Consider a case where the SCC receives a sequence of eight bytes, called A, B, C, D, E, F, G and H, with A received first. Now suppose that A is the sync character, the CRC is calculated on B, C, E, and F, and that F is the last byte of this message. This process is used to control the SCC.

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

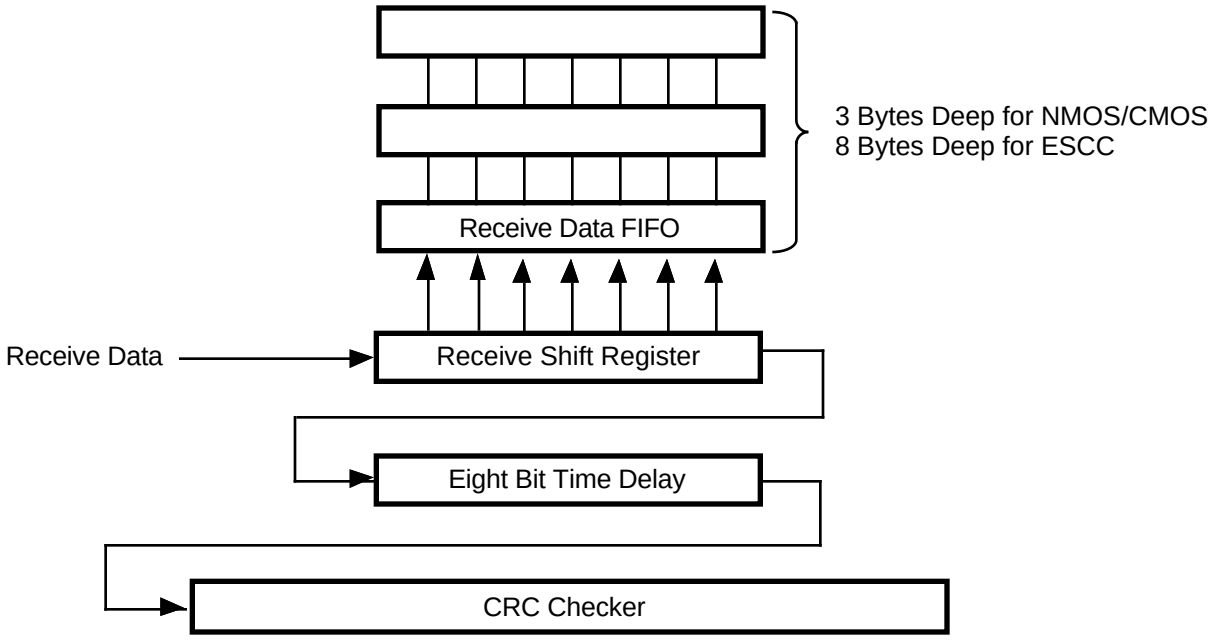



**Figure 4-9. Receive CRC Data Path**

Before A is received, the receiver is in Hunt mode and the CRC is disabled. When A is in the receive shift register, it is compared with the contents of WR7. Since A is the sync character, the bit patterns match and receive leaves Hunt mode, but character A is not transferred to the receive data FIFO.

After eight-bit times, B is loaded into the receive data FIFO. The CRC remains disabled even though somewhere during the next eight bit times the processor reads B and enables the CRC. At the end of this eight-bit time, B is in the 8-bit delay and C is in the receive shift register.

Character C is loaded into the receive data FIFO and at the same time the CRC checker becomes enabled. During the next eight-bit time, the processor reads C and since the CRC is enabled within this period, the SCC has calculated the CRC on B, character C is the 8-bit delay, and D is in the Receive Shift register. D is then loaded into the receive data FIFO and at some point during the next eight-bit time the processor reads D and disables the CRC. At the end of these eight-bit times, the CRC has been calculated on C, character D is in the 8-bit delay, and E is in the Receive Shift register.

Now E is loaded into the receive data FIFO. During the next eight-bit time, the processor reads E and enables the CRC. During this time E shifts into the 8-bit delay, F enters the Receive Shift register and the CRC is not being calculated on D. After these eight-bit times have elapsed, E is in the 8-bit delay, and F is in the Receive Shift register. Now F is transferred to the receive data FIFO and the CRC is enabled. During the next eight-bit times, the processor reads F and leaves the CRC enabled. The processor detects that this is the last character in the message and prepares to check the result of the CRC computation. However, another sixteen bit-times are required before the CRC has been calculated on all of character F.

At the end of eight-bit times, F is in the 8-bit delay and G is in the Receive Shift register. At this time, it is transferred to the receive data FIFO. Character G is read and discarded by the processor. Eight-bit times later, H is also transferred to the receive data FIFO. The result of a CRC calculation is latched in to the Receive Error FIFO at the same time as data is written to the Receive Data FIFO. Thus, the CRC result through character F accompanies character H in the FIFO and will be valid in RR1 until character H is read from the Receive Data FIFO. The CRC checker is disabled and reset at any time after character H is transferred to the Receive Data FIFO. Recall, however, that internally the CRC is not disabled until after this occurs. A better alternative is to place the receiver in Hunt mode, which automatically disables and resets the CRC checker. See Table 4-7 for a condensed description.

**Modem Controls.** Up to two modem control signals associated with the receiver are available in Synchronous modes: /DTR//REQ and /DCD. The /DTR//REQ pin carries the inverted state of the DTR bit (D7) in WR5 unless this pin has been programmed to carry a DMA Request on Transmit signal. The /DCD pin is ordinarily a simple input to the DCD bit in RR0. However, if the Auto Enables mode is selected by setting D5 of WR3 to 1, this pin becomes an enable for the receiver. Therefore, if Auto Enables is ON and the /DCD pin is High, the receiver is disabled; while the /DCD pin is Low, the receiver is enabled.

Note that with Auto Enables mode enabled, when /DCD goes inactive, the receiver stops immediately and the character being assembled is lost.

**Initialization.** The initialization sequence for the receiver in character-oriented mode is WR4 first, to select the mode, then WR10 to modify it if necessary; WR6 and WR7 to program the sync characters; WR3 and WR5 to select the various options. At this point the other registers are initialized as necessary. When all this is completed, the receiver is enabled by setting bit D0 of WR3 to a one. A summary is shown in Table 4-8. A detailed example of using the SCC in 16-bit sync mode is available in the application note "SCC in Binary Synchronous Communications."

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

**Table 4-7. Enabling and Disabling CRC**

| A | B | C | D | E | F | G | H |
|---|---|---|---|---|---|---|---|
| (Sync) | (Data1) | (Data2) | (Data3) | (CRC1) | (CRC2) | (Data) | (Data) |

Note: No CRC Calculation on "D"

| Stage | Direction of Data Coming into SCC → | Shift Register | Receive Data FIFO | Delay Register | CRC | Notes |
|---|---|---|---|---|---|---|
| 1 | H G F E D C B | | | | d | |
| | H G F E D C | A | | | d | |
| 2 | H G F E D<br>CPU Read<br>CPU Enables CR | B | B | | d | |
| 3 | H G F E<br>CPU Read | C | C | B | e | |
| | H G F<br>CPU Read<br>CPU Disables CR | D | D* | C | e | CRC Calc on B<br>CRC Calc on C |
| 4 | H G<br>CPU Read<br>CPU Enables CR | E | E | D | d | |
| | H G<br>CPU Read | F | F | E | e | CRC Calc is Disabled on D<br>CRC Calc on E |
| 5 | H<br>CPU Reads & Disa | G | G | F | e | |
| | | H | | G | e | CRC Calc on F<br>CRC Calc on F |
| | Read RR1 D<br>Read H & Disca | | H<br>H | | | Result latched in Error FIFO † |

**Legend:**

* Usually D is a end-of-message character indicator.

† The status is latched on the Error FIFO for each received byte. In the calculation of F, the CRC error flag in the Error FIFO will be 0 for an error free message.

d = disabled

e = enabled

**A B C D E F G H**
A = SYNC
B - F = Data with E = CRC1 and F = CRC2
G and H are arbitrary data (Pad Character)

**Table 4-8. Initializing the Receiver in Character-Oriented Mode**

| | Bit Number | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Reg** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **Description** |
| WR4 | 0 | 0 | 0 | x | 0 | 0 | 0 | 0 | Select x1 clock, enable sync mode, & no parity<br>x=0 for 8-bit sync, x=1 for 16-bit sync |
| WR3 | r | x | 0 | 1 | 1 | 0 | 0 | 0 | rx=# of Rx bits/char, No auto enable, enter Hunt,<br>Enable Rx CRC, No sync character load inhibit |
| WR5 | d | t | x | 0 | 0 | 0 | r | 1 | d=inverse state of DTR pin, tx=# of Tx bits/char,<br>use CRC-16, r=inverse state of /RTS pin, CRC enable |
| WR6 | x | x | x | x | x | x | x | x | sync character, lower byte |
| WR7 | x | x | x | x | x | x | x | x | sync character, upper byte |
| WR10 | c | 0 | 0 | 0 | i | 0 | 0 | s | c=CRC preset, NRZ data, i=idle line condition<br>s=size of sync character |
| WR3 | r | x | 0 | 1 | 1 | 0 | 0 | 1 | Enable Receiver |
| WR5 | d | t | x | 0 | 1 | 0 | r | 1 | Enable Transmitter |
| WR0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Reset CRC generator |

### 4.3.3 Transmitter/Receiver Synchronization

The SCC contains a transmitter-to-receiver synchronization function that is used to guarantee that the character boundaries for the received and transmitted data are the same. In this mode, the receiver is in Hunt and the transmitter is idle, sending either all 1s or all 0s. When the receiver recognizes a sync character, it leaves Hunt mode; one character time later the transmitter is enabled and begins sending sync characters. Beyond this point the receiver and transmitter are again completely independent, except that the character boundaries are now aligned (Figure 4-10).

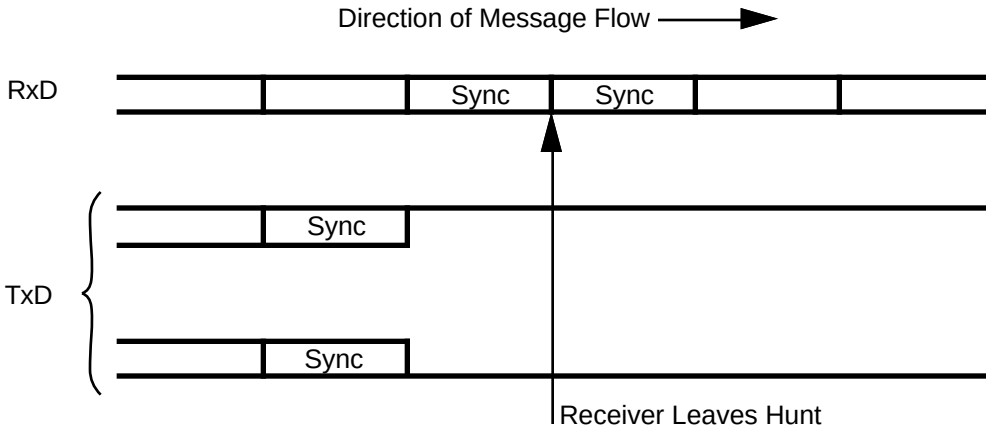



**Figure 4-10. Transmitter to Receiver Synchronization**

There are several restrictions on the use of this feature in the SCC. First, it only works with 6-bit, 8-bit or 16-bit sync characters. The data character length for both the receiver and the transmitter must be six bits with 6-bit sync character, and eight bits with an 8-bit or 16-bit sync character. Of course, the receive and transmit clocks must have the same rate as well as the proper phase relationship.

A specific sequence of operations must be followed to synchronize the transmitter to the receiver. Both the receiver and transmitter must have been initialized for operation in Synchronous mode sometime in the past, although this initialization need not be redone each time the transmitter is synchronized to the receiver. The transmitter is disabled by setting bit D3 of WR5 to 0. At this point the transmitter will send continuous 1s. If it is required that continuous

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

0s be transmitted, the Send Break bit (D4) in WR5 is set to 1. The transmitter is now idling but is still placed in the transmitter to receiver synchronization mode. This is accomplished by setting the Loop Mode bit (D1) in WR10 and then enabling the transmitter by setting bit D3 of WR5 to 1. At this point, the processor should set the Go Active on Poll bit (D4) in WR10. The final step is to force the receiver to search for sync characters. If the receiver is currently disabled, the receiver enters Hunt mode when it is enabled, by setting bit D0 of WR3 to 1. If the receiver is already enabled, it is placed in Hunt mode by setting bit D4 of WR3 to 1. Once the receiver leaves Hunt mode, the transmitter is activated on the following character boundary.

## 4.4 BIT-ORIENTED SYNCHRONOUS (SDLC/HDLC) MODE

Synchronous Data Link Control mode (SDLC) uses synchronization characters similar to Bisync and Monosync modes (such as flags and pad characters). It is a bit-oriented protocol instead of a byte-oriented protocol. High level Data Link Control (HDLC) is defined as CCITT, also EIAJ and other standards; SDLC is one of the implementations made by IBM®. The SDLC protocol uses the technique of zero insertion to make all data transparent from SYNC characters. All references to SDLC in this manual apply to both SDLC and HDLC.

The basic format for SDLC is a frame (Figure 4-11). A Frame is marked at the beginning and end by a unique flag pattern. The flags enclose an address, control, information, and frame check fields. There are many different implementations of the SDLC protocol and many do not use all of the fields. The SCC provides many features to control how each of the fields is received and transmitted.

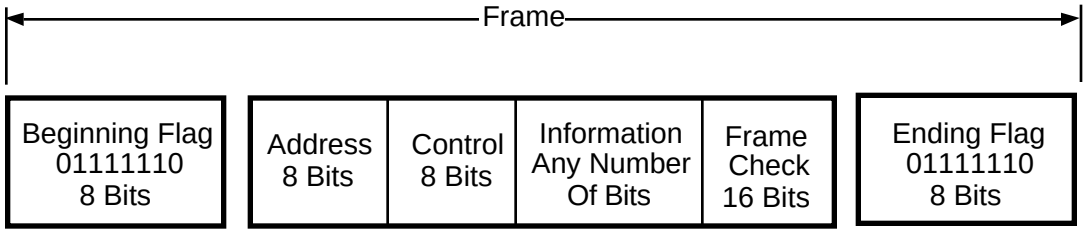



**Figure 4-11. SDLC Message Format**

Frames of information are enclosed by a unique bit pattern called a flag. The flag character has a bit pattern of "01111110" (7E Hex). This sequence of six consecutive ones is unique because all data between the opening and closing flags is prohibited from having more than five consecutive 1s. The transmitter guarantees this by watching the transmit data stream and inserting a 0 after five consecutive 1s, regardless of character boundaries. In turn, the receiver searches the receive data stream for five consecutive 1s and deletes the next bit if it is a 0. Since the SDLC mode does not use characters of defined length, but rather works on a bit-by-bit basis, the 01111110 flag can be recognized at any time. Inserted and removed 0s are not included in the CRC calculation. Since the transmission of the flag character is excluded from the zero insertion logic, its transmission is guaranteed to be seen as a flag by the receiver. The zero insertion and deletion is completely transparent to the user.

Because of the zero insertion/deletion, actual bit length on the transmission line may be longer than the number of bits sent.

The two flags that delineate the SDLC frame serve as reference points when positioning the address and control fields, and they initiate the transmission error check. The ending flag indicates to the receiving station that the 16-bits just received constitute the frame check (CRC; also referred to as FCS or Frame Check Sequence). The ending flag can be followed by another frame, another flag, or an idle. This means that when two frames follow one another, the intervening flag may simultaneously be the ending flag of the first frame and the beginning flag of the next frame. This case is usually referred to as "Back-to-Back Frames".

The SCC's SDLC address field is eight bits long and is used to designate which receiving stations accept a transmitted message. The 8-bit address allows up to 254 (00000001 through 11111110) stations to be addressed uniquely or a global address (11111111) is used to broadcast the message to all stations. Address 0 (00000000) is usually used as a Test packet address.

The control field of a SDLC frame is typically 8 bits, but can be any length. The control field is transparent to the SCC

and is treated as normal data by the transmit and receive logic.

The information field is not restricted in format or content and can be of any reasonable length (including zero). Its maximum length is that which is expected to arrive at the receiver error-free most of the time. Hence, the determination of maximum length is a function of the communication channel's error rate. Usually the upper layer of the protocol specifies the packet size. Although the data is always written/read in a given character size, the Residue Code feature provides the mechanism to read any number of bits at the end of the frame that do not make up a full character. This allows for the data field to be an arbitrary number of bits long.

The frame check field is used to detect errors in the received address, control and information fields. The method used to test if the received data matches the transmitted data, is called a Cyclic Redundancy Check (CRC). The SCC has an option to select between two CRC polynomials, and in SDLC mode only the CRC-CCITT polynomial is used because the transmitter in the SCC automatically inverts the CRC before transmission. To compensate for this, the receiver checks the CRC result for the bit pattern 0001110100001111. This is consistent with bit-oriented protocols such as SDLC, HDLC, and ADCCP and the others.

There are two unique bit patterns in SDLC mode besides the flag sequence. They are the Abort and EOP (End of Poll) sequence. An Abort is a sequence of seven to thirteen consecutive 1s and is used to signal the premature termination of a frame. The EOP is the bit pattern 11111110, which is used in loop applications as a signal to a secondary station that it may begin transmission.

SDLC mode is selected by setting bit D5 of WR4 to 1 and bits D4, D3, and D2 of WR4 to 0. In addition, the flag sequence is written to WR7. Additional control bits for SDLC mode are located in WR10 and WR7' (85X30).

### 4.4.1 SDLC Transmit

In SDLC mode, the transmitter moves characters from the transmitter buffer (on the ESCC, four-byte transmitter FIFO) to the Transmit Shift register, through the zero inserter and out to the TxD pin. The insertion of zero is completely transparent to the user. Zero insertion is done to all transmitted characters except the flag and abort.

A SDLC frame must have the 01111110 (7E Hex) flag sequence transmitted before the data. This is done automatically by the SCC by programming WR7 with 7EH as part of the device initialization, enabling the transmitter, and then writing data. If the SCC is programmed to idle Mark (WR10 D3=1), special consideration must be taken to transmit the opening flag. Ordinarily, it is necessary to reset the WR10 D3 to idle flag, wait 8-bit times, and then write data to the transmitter. It is necessary to wait eight bit times before writing data because '1s' are transmitted eight at a time and all eight must leave the Transmit Shift register before a flag is loaded.

The ESCC has two improvements over the NMOS/CMOS version to control the transmission of the flag at the beginning of a frame. Additionally, the ESCC has improved features to ease the handling of SDLC mode of operation, including a function to deactivate the /RTS signal at the end of the packet automatically. For these features, refer to the next subsection, 4.4.1.2, "ESCC Enhancements for SDLC Transmit."

The number of bits per transmitted character is controlled by bits D6 and D5 of WR5 and the way the data is formatted within the transmit buffer. The bits in WR5 allow the option of five, six, seven, or eight bits per character. In all cases, the data must be right justified, with the unused bits being ignored, except in the case of five bits per character. When five bits per character are selected, the data may be formatted before being written to the transmit buffer. This allows transmission of one to five bits per character (Table 4-2).

An additional bit, carrying parity information, is automatically appended to every transmitted character by setting bit D0 of WR4 to 1. This bit is sent in addition to the number of bits specified in WR4 or by the data format. The parity sense is selected by bit D1 of WR4. Parity is not normally used in SDLC mode as the overhead of parity is unnecessary due to the availability of the CRC.

The SCC transmits address and control fields as normal data and does not automatically send any address or control information. The value programmed into WR6 is used by the receiver to compare the address of the received frame (if address search mode is enabled), but WR6 is not used by the transmitter. Therefore, the address is written to the transmitter as the first byte of data in the frame.

The information field can be any number of characters long. On the NMOS/CMOS version, the transmitter can interrupt the CPU when the transmit buffer is empty. On the ESCC, the transmitter can interrupt the CPU when the entry location of the Transmit FIFO is empty or when the Transmit FIFO is completely empty. Also, the NMOS/CMOS version can issue a DMA request when the transmit buffer is empty, while the ESCC can issue a DMA request when the entry location of the Transmit FIFO is empty or when the Transmit FIFO is completely empty. This allows the ESCC user to optimize the response to the application requirements. Since the ESCC has a four byte Transmit FIFO buffer, the Transmit Buffer Empty (TBE) bit (D2 of RR0) will become set when the entry location of the Transmit FIFO becomes empty. The TBE bit will reset when a byte of data is loaded into the entry location of the Transmit FIFO. For more details on this subject, refer to

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

Section 2.4.8 "Transmit Interrupts and Transmit Buffer Empty bit".

The character length may be changed on the fly, but the desired length must be selected before the character is loaded into the Transmit Shift register from the transmit data FIFO. The easiest way to ensure this is to write to WR5 to change the character length before writing the data to the transmit buffer. Note that although the character can be any length, most protocols specify the address/control field as 8-bit fields. The SCC receiver checks the address field as 8-bit, if address search mode is enabled.

Only the CRC-CCITT polynomial is used in SDLC mode. This is selected by setting bit D2 in WR5 to 0. This bit controls the selection for both the transmitter and receiver. The initial state of the generator and checker is controlled by bit D7 of WR10. When this bit is set to 1, both the generator and checker have an initial value of all 1s, and if this bit is set to 0, the initial values are all 0s.

The SCC does not automatically preset the CRC generator, so this is done in software. This is accomplished by issuing the Reset Tx CRC command, which is encoded in bits D7 and D6 of WR0. For proper results, this command is issued while the transmitter is enabled and idling. If the CRC is to be used, the transmit CRC generator is enabled by setting bit D0 of WR5 to 1. The CRC is normally calculated on all characters between opening and closing flags, so this bit is usually set to 1 at initialization and never changed. On the 85X30 with Auto EOM Latch reset mode enabled (WR7' bit D1=1), resetting of the CRC generator is done automatically.

Enabling the CRC generator is not sufficient to control the transmission of the CRC. In the SCC, this function is controlled by Tx Underrun/EOM bit, which may be reset by the processor and set by SCC. On the 85X30 with Auto EOM Reset mode enabled (WR7' bit D1=1), resetting of the Tx Underrun/EOM Latch is done automatically.

Ordinarily, a frame is terminated with a CRC and a flag, but the SCC may be programmed to send an abort and a flag in place of the CRC. This option allows the SCC to abort a frame transmission in progress if the transmitter is accidentally allowed to underrun. This is controlled by the Abort/Flag on Underrun bit (D2) in WR10. When this bit is set to 1, the transmitter will send an abort and a flag in place of the CRC when an underrun occurs. The frame is terminated normally with a CRC and a flag if this bit is 0.

The SCC is also able to send an abort by a command from the processor. When the Send Abort command is issued in WR0, the transmitter sends eight consecutive 1s and then idles. Since up to five consecutive 1s may be sent prior to the command being issued, a Send Abort causes a sequence of from eight to thirteen 1s to be transmitted. The Send Abort command also clears the transmit data FIFO.

When transmitting in SDLC mode, note that all data passes through the zero inserter, which adds an extra five bit times of delay between the Transmit Shift register and the TxD Pin.

When the transmitter underruns (both the Transmit FIFO and Transmit Shift register are empty), the state of the Tx Underrun/EOM bit determines the action taken by the SCC.

If the Tx Underrun/EOM bit is set to 1 when the underrun occurs, the transmitter sends flags without sending the CRC. If this bit is reset to 0 when the underrun occurs, the transmitter sends either the accumulated CRC followed by flags, or an abort followed by flags, depending on the state of the Abort/Flag on the Underrun bit in the WR10, bit D1. A summary is shown in Table 4-9.

The Reset Tx Underrun/EOM Latch command is encoded in bits D7 and D6 of WR0.

**Table 4-9. ESCC Action Taken on Tx Underrun**

| Tx Underrun /EOM Latch Bit | Abort/Flag | Action taken by ESCC upon transmit underrun |
|---|---|---|
| 0 | 0 | Sends CRC followed by flag |
| 0 | 1 | Sends abort followed by flag |
| 1 | x | Sends flag |

The SCC sets the Tx Underrun/EOM latch when the CRC or abort is loaded into the shift register for transmission. This event can cause an interrupt, and the status of the Tx Underrun/EOM latch can be read in RR0.

Resetting the Tx Underrun/EOM latch is done by the processor via the command encoded in bits D7 and D6 of WR0. On the 85X30, this also can be accomplished by setting WR7' bit D1 for Auto Tx Underrun/EOM Latch Reset mode enabled. For correct transmission of the CRC at the end of a frame, this command must be issued after the first character is written to the SCC but before the transmitter underruns after the last character written to the SCC. The command is usually issued immediately after the first character is written to the SCC so that the abort or CRC is sent if an underrun occurs inadvertently. The Abort/Flag on Underrun bit (D2) in WR10 is usually set to 1 at the same time as the Tx Underrun/EOM bit is reset so that an abort is sent if the transmitter underruns. The bit is then set to 0

near the end of the frame to allow the correct transmission of the CRC.

In this paragraph the term "completely sent" means shifted out of the Transmit Shift register, not shifted out of the zero inserter, which is an additional five bit times of delay. In SDLC mode, if the transmitter is disabled during transmission of a character, that character will be "completely sent." This applies to both data and flags. However, if the transmitter is disabled during the transmission of the CRC, the 16-bit transmission will be completed, but the remaining bits are from the Flag register rather than the remainder of the CRC.

The initialization sequence for the transmitter in SDLC mode is:

1. WR4 selects the mode.
2. WR10 modifies it if necessary.
3. WR7 programs the flag.
4. WR3 and WR5 selects the various options.

At this point the other registers should be initialized as necessary. When all of this is complete, the transmitter may be enabled by setting bit D3 of WR5 to 1. Now that the transmitter is enabled, the CRC generator may be initialized by issuing the Reset Tx CRC Generator command in WR0.

#### 4.4.1.1 Modem Control signals related to SDLC Transmit

There are two modem control signals associated with the transmitter provided by the SCC. The /RTS pin is a simple output that carries the inverted state of the RTS bit (D1) in WR5. The /CTS pin is ordinarily a simple input to the CTS bit in RR0. However, if Auto Enables mode is selected, this pin becomes an enable for the transmitter. If Auto Enables is on and the /CTS pin is High, the transmitter is disabled. The transmitter is enabled if the /CTS pin is Low.

#### 4.4.1.2 ESCC Enhancements for SDLC Transmit

The ESCC has the following enhancements available in the SDLC mode of operation which can reduce CPU overhead dramatically. These features are:

- Deeper Transmit FIFO (Four Bytes)
- CRC takes priority over the data
- Auto EOM Reset (WR7' bit D1)
- Auto Tx Flag (WR7' bit D0)
- Auto RTS Deactivation (WR7' bit D2)
- TxD pin forced High after closing flag in NRZI mode

**Deeper Transmit FIFO:** The ESCC has a four byte deep Transmit FIFO, where the NMOS/CMOS version has a one byte deep transmit buffer. To maximize the system's performance, there are two modes of operation for the transmit interrupt and DMA request, which are programmed by bit D5 of WR7'.

The ESCC sets WR7' bit D5 to 1 following a hardware or software reset. This is done to provide maximum compatibility with existing SCC designs. In this mode, the ESCC generates the transmit buffer empty interrupt and DMA transmit request when the Transmit FIFO is completely empty. Interrupt driven systems can maximize efficiency by writing four bytes for each entry into the Transmit Interrupt Service Routine (TISR), filling the Transmit FIFO without having to check any status bits. Since the TBE status bit is set if the entry location of the FIFO is empty, this bit can be tested at any time if more data is written. Applications requiring software compatibility with the NMOS/CMOS version can test the TBE bit in the TISR after each data write to determine if more data can be written. This allows a system with an ESCC to minimize the number of transmit interrupts, but not overflow SCC systems. DMA driven systems originally designed for the SCC can use this mode to reassert the DMA request for more data after the first byte written to the FIFO is loaded to the Transmit Shift register. Consequently, any subsequent reassertion allows the DMA sufficient time to detect the High-to-Low edge.

If WR7' D5 is reset to 0, the transmit buffer empty interrupt and DMA request are generated when the entry location of the FIFO is empty. Therefore, if more than one byte is required to fill the entry location of the FIFO, the ESCC generates interrupts or DMA requests until the entry location of the FIFO is filled. The transmit DMA request pin (either /WAIT//REQ or /DTR//REQ) goes inactive after each data transfer, then goes active again and, consequently, generates a High-to-Low edge for each byte. Edge triggered DMA should be enabled before the transmit DMA function is enabled in the ESCC to guarantee that the ESCC does not generate the edge before the DMA is ready.

**CRC takes priority over data:** On the NMOS/CMOS version, the data has higher priority over CRC data. Writing data before the Tx interrupt, after loading the closing flag into the Transmit Shift register, terminates the packet illegally. In this case, CRC byte(s) are replaced with Flag or Sync patterns, followed by the data written. On the ESCC, CRC has priority over the data. Consequently, after the Underrun/EOM (End of message) interrupt occurs, the ESCC accepts the data for the next packet without fear of collapsing the packet. On the ESCC, if data was written during the time period described above, the TBE bit (bit D2 of RR0) is NOT set; even if the 2nd TxIP is guaranteed to set when the flag/sync pattern is loaded into the Transmit Shift register (Section 2.4.8). For the detailed timing on this, refer to Figures 2-17 and 2-18.

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

Hence, on the ESCC, there is no need to wait for the 2nd TxIP bit to set before writing data for the next packet which reduces the overhead.

**Auto EOM Reset (WR7' bit D1):** As described above, the Tx Underrun/EOM Latch has to be reset before the Transmit Shift register completes shifting out the last character, but after first character has been written. One of the ways to reset it is for the CPU to issue the "Reset Tx Underrun/EOM Latch" command. The other method to accomplish it is by the "Automatic EOM Latch Reset feature" by setting bit D1 in WR7', which is one of the enhancements made to the ESCC. By setting this bit to one, it eliminates the need for the CPU command. In this mode, the CRC generator is automatically reset at the start of every packet, without the CPU command. Hence, it is not required to reset the CRC generator prior to writing data into the ESCC. This is particularly valuable to a DMA driven system where issuing CPU commands while the DMA is transferring data is difficult. Also, it is very useful if the data rate is very high and the CPU may not be able to issue the command on time.

**Auto Tx Flag (WR7' bit D0):** With the NMOS/CMOS version of the SCC, in order to accomplish Mark idle, it is required to enable the transmitter as Mark idle; then re-program to Flag idle before writing first data, and then reprogram again to mark idle as described above. Normally, during mark idle, the transmitter sends continuous flags, but the ESCC can idle MARK under program control. By setting the Mark/Flag idle bit (D3) in WR10 to 1, the transmitter sends continuous 1s in place of the idle flags. The closing flag always transmits correctly even when this mode is selected. Normally, it is necessary to reset WR10 D3 to 0 before writing data for the next frame. However, on the ESCC, if WR7' bit D0 is set to 1, an opening flag is transmitted automatically and it is not necessary for the CPU to turn the Mark Idle feature on and off between frames.

**Note:** When this mode in not in effect (WR7' D0=0), the Mark/Flag idle bit is clear to 0, allowing a flag to be transmitted before data is written to the transmit buffer. Care must be exercised in doing this because the continuous 1s are transmitted eight at a time and all eight must leave the Transmit Shift register. This allows a flag to be loaded into it before the first data is written to the Transmit FIFO.

**Auto RTS Deactivation (WR7' bit D2):** Some applications require toggling the modem signal to indicate the end of the packet. With the NMOS/CMOS version, this requires intensive CPU support; the CPU needs time to determine whether or not the last bit of the closing flag has left the TxD pin. The ESCC has a new feature to deactivate the /RTS signal when the last bit of the closing flag clears the TxD pin.

If this feature is enabled by setting bit D2 of WR7', and when WR5 bit D1 is reset during the transmission of a SDLC frame, the deassertion of the /RTS pin is delayed until the last bit of the closing flag clears the TxD pin. The /RTS pin is deasserted after the rising edge of the transmit clock cycle on which the last bit of the closing flag is transmitted. This implies that the ESCC is programmed for Flag on Underrun (WR10 bit D2=1) for the /RTS pin to deassert at the end of the frame. (Otherwise, the deassertion occurs when the next flag is transmitted). This feature works independently of the programmed transmitter idle state. In Synchronous modes other than SDLC, the /RTS pin immediately follows the state programmed into WR5 D1. Note that if the /RTS pin is connected to one of the general purpose inputs (/CTS or /DCD), it can be used to generate an external status interrupt when a frame is completely transmitted.

**NRZI forced High after closing flag:** On the CMOS/NMOS version of the SCC in the SDLC mode of operation with NRZI mode of encoding and mark idle (WR10 bit D6=0, D5=1, D3=1), the state of the TxD pin after transmission of the closing flag is undetermined, depending on the last data sent. With the ESCC in the same operation mode (SDLC, NRZI, with mark idle), the TxD pin is automatically forced High on the falling edge of the TxC of the last bit of the closing flag, and then the transmitter goes to the mark idle state.

There are several different ways for a transmitter to go into the idle state. In each of the following cases, the TxD pin is forced High when the mark idle condition is reached; data, CRC (2 bytes), flag and idle; data, flag and idle; data, abort (on underrun) and idle; data, abort (by command) and idle; idle, flag and command to idle mark. The force High feature is disabled when the mark idle bit is reset (programmed as mark idle). This feature is used in combination with the automatic SDLC opening flag transmission feature, WR7' bit D0=1, to assure that data packets are properly formatted. When these features are used together, it is not necessary for the CPU to issue any commands after sending a closing flag in combination with NRZI data encoding. (On the NMOS/CMOS version, this is accomplished by channel reset, followed by re-initializing the channel). If WR7' bit D0 is reset, like in the NMOS/CMOS version, it is necessary to reset the mark idle bit (WR10, bit D3) to enable flag transmission before a SDLC packet is transmitted.

### 4.4.2 SDLC Receive

The receiver in the SCC always searches the receive data stream for flag characters in SDLC mode. Ordinarily, the receiver transfers all received data between flags to the receive data FIFO. However, if the receiver is not in Hunt mode no data is received. The receiver is in Hunt mode when first enabled, or the receiver is placed in Hunt mode

by the processor issuing the Enter Hunt mode command in WR3. This bit (D4) is a command, and writing a 0 to it has no effect. The Hunt status of the receiver is reported by the Sync/Hunt bit in RR0.

Sync/Hunt is one of the possible sources of external/status interrupts, with both transitions causing an interrupt. This is true even if the Sync/Hunt bit is set as a result of the processor issuing the Enter Hunt mode command.

The receiver automatically enters Hunt mode if an abort is received. Because the receiver always searches the receive data stream for flags, and automatically enters Hunt Mode when an abort is received, the receiver always handles frames correctly. The Enter Hunt Mode command should never be needed. The SCC drives the /SYNC pin Low to signal that a flag has been recognized. The timing for the /SYNC signal is shown in Figure 4-12.

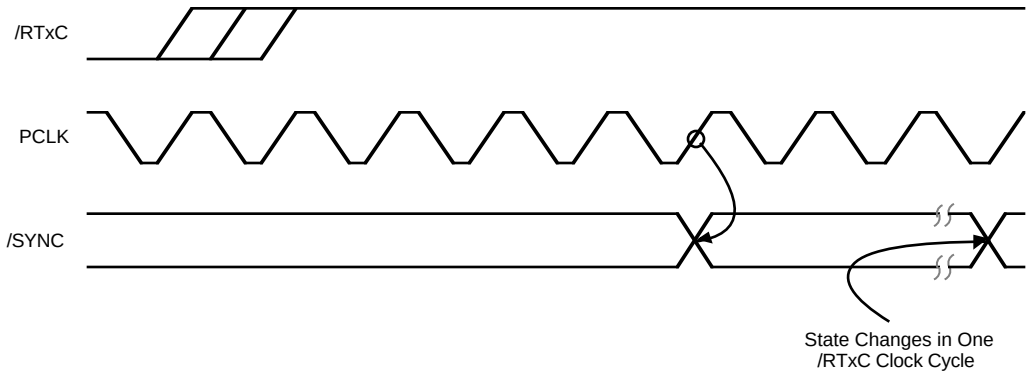



**Figure 4-12. /SYNC as an Output**

The SCC assumes the first byte in an SDLC frame is the address of the secondary station for which the frame is intended. The SCC provides several options for handling this address.

If the Address Search Mode bit (D2) in WR3 is set to 0, the address recognition logic is disabled and all received frames are transferred to the receive data FIFO. In this mode the software must perform any address recognition.

If the Address Search Mode bit is set to 1, only those frames whose address matches the address programmed in WR6 or the global address (all 1s) will be transferred to the receive data FIFO.

The address comparison is across all eight bits of WR6 if the Sync Character Load inhibit bit (D1) in WR3 is set to 0. The comparison may be modified so that only the four most significant bits of WR6 match the received address. This mode is selected by setting the Sync Character Load inhibit bit to 1. In this mode, however, the address field is still eight bits wide. The address field is transferred to the receive data FIFO in the same manner as data. It is not treated differently than data.

The number of bits per character is controlled by bits D7 and D6 of WR3. Five, six, seven, or eight bits per character may be selected via these two bits. The data is right-justified in the receive buffer. The SCC merely takes a snapshot of the receive data stream at the appropriate times, so the "unused" bits in the receive buffer are only the bits following the character.

An additional bit carrying parity information is selected by setting bit D6 of WR4 to 1. This also enables parity in the transmitter. The parity sense is selected by bit D1 of WR4. Parity is not normally used in SDLC mode.

The character length can be changed at any time before the new number of bits have been assembled by the receiver. Care should be exercised, however, as unexpected results may occur. A representative example, switching from five bits to eight bits and back to five bits, is shown in Figure 4-13.

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

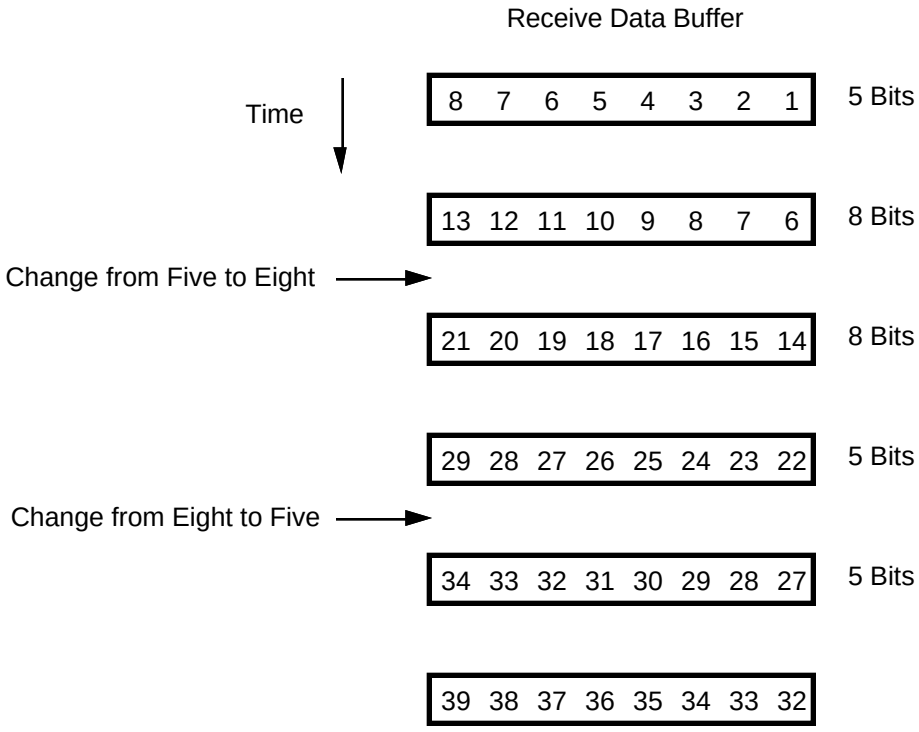



**Figure 4-13. Changing Character Length**

Most bit-oriented protocols allow an arbitrary number of bits between opening and closing flags. The SCC allows for this by providing three bits of Residue Code in RR1. These indicate which bits in the last three bytes transferred from the receive data FIFO by the processor are actually valid data bits (and not part of the frame check sequence or CRC). Table 4-10 gives the meanings of the different codes for the four different character length options. The valid data bits are right-justified, meaning, if the number of valid bits given by the table is less than the character length, then the bits that are valid are the right-most or least significant bits. It should also be noted that the Residue Code is only valid at the time when the End of Frame bit in RR1 is set to 1.

**Table 4-10. Residue Codes**

| Residue Code | | | Bits in Previous Byte | | | | Bits in Second Previous Byte | | | | Bits in Third Previous Byte | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **2** | **1** | **0** | **8B/C** | **7B/C** | **6B/C** | **5B/C** | **8B/C** | **7B/C** | **6B/C** | **5B/C** | **8B/C** | **7B/C** | **6B/C** | **5B/C** |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 3 | 1 | 0 | 0 | 8 | 7 | 5 | 2 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 4 | 2 | 0 | 0 | 8 | 7 | 6 | 3 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 3 | 1 | 0 | 8 | 7 | 6 | 4 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 6 | 4 | 2 | 0 | 8 | 7 | 6 | 5 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 7 | 5 | 3 | 1 | 8 | 7 | 6 | 5 |
| 0 | 1 | 1 | 0 | 0 | 0 | | 8 | 6 | 4 | | 8 | 7 | 6 | |
| 1 | 1 | 1 | 1 | 0 | | | 8 | 7 | | | 8 | 7 | | |
| 0 | 0 | 0 | 2 | | | | 8 | | | | 8 | | | |

As indicated in the table, these bits allow the processor to determine those bits in the information (and not CRC) field. This allows transparent retransmission of the received frame. The Residue Code bits do not go through a FIFO, so they change in RR1 when the last character of the frame is loaded into the receive data FIFO. If there are any characters already in the receive data FIFO the Residue Code is updated before they are read by the processor.

As an example of how the codes are interpreted, consider the case of eight bits per character and a residue code of 101. The number of valid bits for the previous, second previous, and third previous bytes are 0, 7, and 8, respectively. This indicates that the information field (I-field) boundary falls on the second previous byte as shown in Figure 4-14.

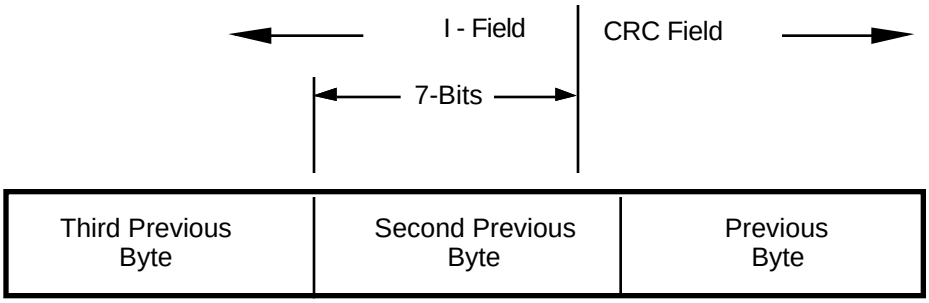



**Figure 4-14. Residue Code 101 Interpretation**

A frame is terminated by the detection of a closing flag. Upon detection of the flag the following actions take place: the contents of the Receive Shift Register are transferred to the receive data FIFO; the Residue Code is latched, the CRC Error bit is latched; the End of Frame upon reaching the top of the FIFO can cause a special receive condition. The processor then reads RR1 to determine the result of the CRC calculation and the Residue Code.

Only the CRC-CCITT polynomial is used for CRC calculations in SDLC mode, although the generator and checker can be preset to all 1s or all 0s. The CRC-CCITT polynomial is selected by setting bit D2 of WR5 to 0. Bit D7 of WR10 controls the preset value. If this bit is set to 1, the generator and checker are preset to 1s, and if this bit is reset, the generator and checker are preset to all 0s.

The receiver expects the CRC to be inverted before transmission, so it checks the CRC result against the value 0001110100001111. The SCC presets the CRC checker whenever the receiver is in Hunt mode or whenever a flag is received, so a CRC reset command is not necessary. However, the CRC checker can be preset by issuing the Reset CRC Checker command in WR0.

The CRC checker is automatically enabled for all data between the opening and closing flags by the SCC in SDLC mode, and the Rx CRC Enable bit (D3) in WR3 is ignored. The result of the CRC calculation for the entire frame is valid in RR1 only when accompanied by the End of Frame bit set in RR1. At all other times, the CRC Error bit in RR1 should be ignored by the processor.

On the NMOS/CMOS version, care must be exercised so that the processor does not attempt to use the CRC bytes that are transferred as data, because not all of the bits are transferred properly. The last two bits of CRC are never transferred to the receive data FIFO and are not recoverable.

On the ESCC, an enhancement has been made allowing the 2nd byte of the CRC to be received completely. This feature is useful when the application requires the 2nd CRC byte as data. For example, applications which operate in transparent mode or protocols using the error checking mechanism other than CRC-CCITT (like 32-bit CRC).

Note the following about SCC CRC operation:

- The normal CRC checking mechanism involves checking over data and CRC characters. If the division remainder is 0, there is no CRC error.
- SDLC is different. The CRC generator, when receiving a correct frame, has a fixed, non-zero remainder. The actual remainder in the receive CRC calculation is checked against this fixed value to determine if a CRC error exists.

A frame is terminated by a closing flag. When the SCC recognizes this flag:

- The contents of the Receive Shift register are transferred to the receive data FIFO.
- The Residue Code is latched, the CRC Error bit is latched in the status FIFO and the End of Frame bit is set in the receive status FIFO.

The End of Frame bit, upon reaching the exit location of the FIFO, will cause a special receive condition. The processor may then read RR1 to determine the result of the CRC calculation as well as the Residue Code. If either the Rx Interrupt on Special Condition Only or the Rx Interrupt on First Character or Special Condition modes are

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

selected, the processor must issue an Error Reset command in WR0 to unlock the Receive FIFO.

In addition to searching the data stream for flags, the receiver in the SCC also watches for seven consecutive 1s, which is the abort condition. The presence of seven consecutive 1s is reported in the Break/Abort bit in RR0. This is one of the possible external/status interrupts, so transitions of this status may be programmed to cause interrupts. Upon receipt of an abort the receiver is forced into Hunt mode where it looks for flags. The Hunt status is also a possible external/status condition whose transition may be programmed to cause an interrupt. The transitions of these two bits occur very close together, but either one or two external/status interrupts may result. The abort condition is terminated when a 0 is received, either by itself or as the leading 0 of a flag. The receiver does not leave Hunt mode until a flag has been received, so two discrete external/status conditions occur at the end of an abort. An abort received in the middle of a frame terminates the frame reception, but not in an orderly manner because the character being assembled is lost.

Up to two modem control signals associated with the receiver are available in SDLC mode:

- The /DTR//REQ pin carries an inverted state of the DTR bit (D7) in WR5 unless this pin has been programmed to carry a DMA Request signal.
- The /DCD pin is ordinarily a simple input to the DCD bit in RR0. However, if the Auto Enables mode is selected by setting bit D5 of WR3 to 1, this pin becomes an enable for the receiver. That is, if Auto Enables is on and the /DCD pin is High, the receiver is disabled. While the /DCD pin is Low, the receiver is enabled.

**SDLC Initialization.** The initialization sequence for SDLC mode is WR4 to select SDLC mode first, WR3 and WR5 to select the various options, WR7 to program flag, and then WR6 for the receive address. At this point the other registers should be initialized as necessary. When all this is completed the receiver is enabled by setting bit D0 of WR3 to a one. A summary is shown in Table 4-11.

**Table 4-11. Initializing in SDLC Mode**

| | Bit # | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Reg** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **Description** |
| WR4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Select x1 clock,<br>SDLC mode, enable sync mode |
| WR3 | r | x | 0 | 1 | 1 | 1 | 0 | 0 | rx=# of Rx bits/char, No auto enable, enter Hunt.<br>Enable Rx CRC, Address Search, No sync character load inhibit |
| WR5 | d | t | x | 0 | 0 | 0 | r | 1 | d=inverse of DTR pin, tx=# of Tx bits/char, use SDLC CRC, r=inverse state of /RTS pin, CRC enable |
| WR7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | SDLC Flag |
| WR6 | x | x | x | x | x | x | x | x | Receiver secondary address |
| WR15 | x | x | x | x | x | x | x | 1 | Enable access to new register |
| WR7' | 0 | 1 | 1 | d | 1 | r | 1 | 1 | Enable extended read, Tx INT on FIFO empty, d=REQUEST timing mode, Rx INT on 4 char, r=RTS deactivation, auto EOM reset, auto flag tx CRC preset to zero, NRZ data,i=idle line |
| WR10 | 0 | 0 | 0 | 0 | i | 0 | 0 | 0 | CRC preset to zero, NRZ data, i=idle line |
| WR3 | r | x | 0 | 1 | 1 | 1 | 0 | 1 | Enable Receiver |
| WR5 | d | t | x | 0 | 1 | 0 | r | 1 | Enable Transmitter |
| WR0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Reset CRC generator |

**Note:** The receiver searches for synchronization when it is in Hunt mode. In this mode, the receiver is idle except for searching the data stream for a flag match.

**Note:** When the receiver detects a flag match it achieves synchronization and interprets the following byte as the address field.

**Note:** The SYNC/HUNT bit in RR0 reports the Hunt Status, and an interrupt is generated upon transitions between the Hunt state and the Sync state.

**Note:** The SCC will drive the /SYNC pin Low for one receive clock cycle to signal that the flag has been received.

### 4.4.3 SDLC Frame Status FIFO

**This feature is not available on the NMOS version.** On the CMOS version and the ESCC, the ability to receive high speed back-to-back SDLC frames is maximized by a 10-bit deep by 19-bit wide status FIFO. When enabled (through WR15, bit D2), it provides a DMA the ability to continue to transfer data into memory so that the CPU can examine the message later. For each SDLC frame, a 14-bit byte count and five status/error bits are stored. The byte count and status bits are accessed through Read Registers 6 and 7. Read Registers 6 and 7 are only accessible when the SDLC FIFO is enabled. The 10x19 status FIFO is separate from the 8-byte Receive Data FIFO.

When the enhancement is enabled, the status in Read Register 1 (RR1) and byte count for the SDLC frame is stored in the 10 x 19 bit status FIFO. This allows the DMA controller to transfer the next frame into memory while the CPU verifies the message was properly received.

Summarizing the operation; data is received, assembled, and loaded into the eight-byte FIFO before being transferred to memory by the DMA controller. When a flag is received at the end of an SDLC frame, the frame byte count from the 14-bit counter and five status bits are loaded into the status FIFO for verification by the CPU. The CRC checker is automatically reset in preparation for the next frame which can begin immediately. Since the byte count and status are saved for each frame, the message integrity can be verified at a later time. Status information for up to 10 frames can be stored before a status FIFO overrun occurs.

If a frame is terminated with an ABORT, the byte count will be loaded to the status FIFO and the counter reset for the next frame.

**FIFO Detail.** For a better understanding of details of the FIFO operation, refer to the block diagram in Figure 4-15.

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

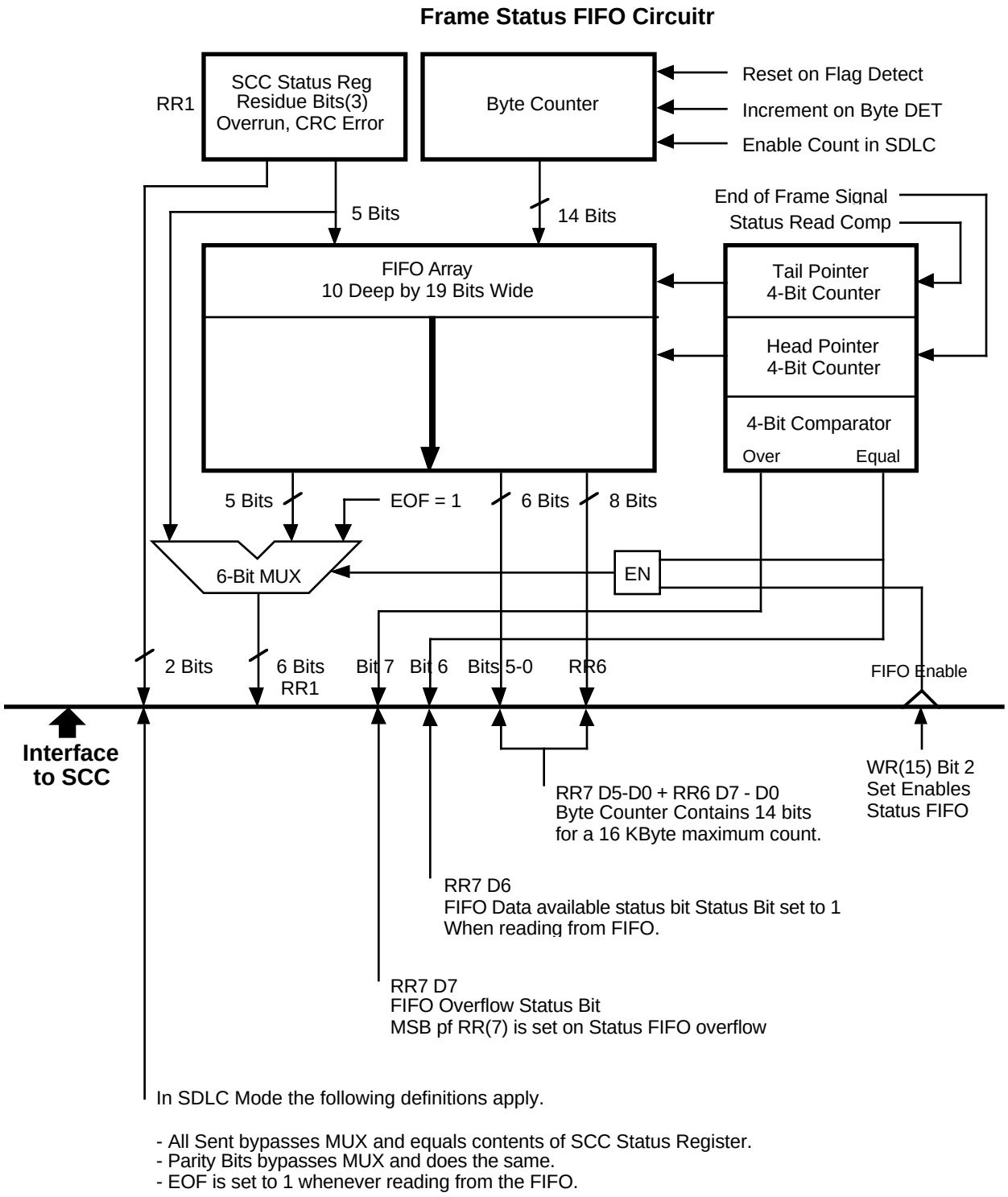



**Figure 4-15. SDLC Frame Status FIFO (N/A on NMOS)**

**Enable/Disable.** The frame status FIFO is enabled when WR15 bit D2 is set and the CMOS/ESCC is in the SDLC/HDLC mode. Otherwise, the status register contents bypass the FIFO and go directly to the bus interface (the FIFO pointer logic is reset either when disabled or via a channel or Power-On Reset). The FIFO mode is disabled on power-up (WR15 D2 is set to 0 on reset). The effects of backward compatibility on the register set are that RR4 is an image of RR0, RR5 is an image of RR1, RR6 is an image of RR2 and RR7 is an image of RR3. For the details of the added registers, refer to Chapter 5. The status of the FIFO Enable signal can be obtained by reading RR15 bit D2. If the FIFO is enabled, the bit is set to 1; otherwise, it is reset.

**Read Operation.** When WR15 bit D2 is set and the FIFO is not empty, the next read to any of status register RR1 or the additional registers RR7 and RR6 is from the FIFO. Reading status register RR1 causes one location of the FIFO to be emptied, so status is read after reading the byte count, otherwise the count is incorrect. Before the FIFO underflows, it is disabled. In this case, the multiplexer is switched to allow status to read directly from the status register, and reads from RR7 and RR6 contain bits that are undefined. Bit D6 of RR7 (FIFO Data Available) is used to determine if status data is coming from the FIFO or directly from the status register, since it is set to 1 whenever the FIFO is not empty.

Since not all status bits are stored in the FIFO, the All Sent, Parity, and EOF bits bypass the FIFO. The status bits sent through the FIFO are Residue Bits (3), Overrun, and CRC Error.

The sequence for proper operation of the byte count and FIFO logic is to read the register in the following order: RR7, RR6, and RR1 (reading RR6 is optional). Additional logic prevents the FIFO from being emptied by multiple reads from RR1. The read from RR7 latches the FIFO empty/full status bit (D6) and steers the status multiplexer to read from the CMOS/ESCC megacell instead of the status FIFO (since the status FIFO is empty). The read from RR1 allows an entry to be read from the FIFO (if the FIFO was empty, logic was added to prevent a FIFO underflow condition).

**Write Operation.** When the end of an SDLC frame (EOF) has been received and the FIFO is enabled, the contents of the status and byte-count registers are loaded into the FIFO. The EOF signal is used to increment the FIFO. If the FIFO overflows, the RR7 bit D7 (FIFO Overflow) is set to indicate the overflow. This bit and the FIFO control logic is reset by disabling and re-enabling the FIFO control bit (WR15 bit 2). For details of FIFO control timing during an SDLC frame, refer to Figure 4-16.

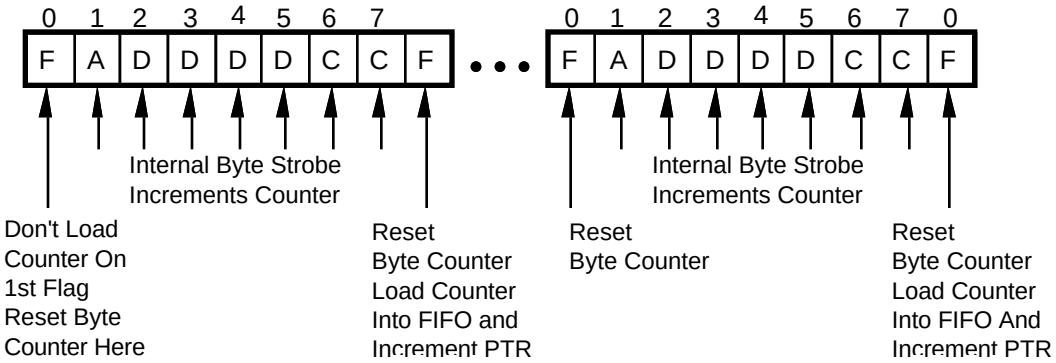



**Figure 4-16. SDLC Byte Counting Detail**

**SDLC Status FIFO Anti-Lock Feature (ESCC only).** When the Frame Status FIFO is enabled and the ESCC is programmed for Special Receive Condition Only (WR1 D4=D3=1), the data FIFO is not locked when a character with End of Frame status is read. When a character with the EOF status reaches the top of the FIFO, an interrupt with a vector for receive data is generated. The command Reset Highest IUS must be issued at the end of the interrupt service routine regardless of whether an interrupt acknowledge cycle had been executed (hardware or software). This allows a DMA to complete a transfer of the received frame to memory and then interrupt the CPU that a frame has been completed without locking the FIFO. Since in the Receive Interrupt on Special Condition Only mode the interrupt vector for receive data is not used, it is used to indicate that the last byte of a frame has been read from the Receive FIFO. This eliminates having to read the frame status (CRC and other status is stored in the status FIFO with the frame byte count).

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

When a character with a special receive condition other than EOF is received (receive overrun, or parity), a special receive condition interrupt is generated after the character is read from the FIFO and the Receive FIFO is locked until the Error Reset command is issued.

### 4.4.4 SDLC Loop Mode

The SCC supports SDLC Loop mode in addition to normal SDLC. SDLC Loop mode is very similar to normal SDLC but is usually used in applications where a point-to-point network is not appropriate (for example, Point-of-Sale terminals). In an SDLC Loop, there is a primary controller that manages the message traffic flow on the loop and any number of secondary stations. In SDLC Loop mode, the SCC operating in regular SDLC mode can act as the primary controller.

A secondary station in an SDLC Loop is always listening to the messages being sent around the loop, and in fact must pass these messages to the rest of the loop by retransmitting them with a one-bit-time delay.

The secondary station can place its own message on the loop only at specific times. The controller signals that secondary stations may transmit messages by sending a special character, called an EOP (End of Poll), around the loop. The EOP character is the bit pattern 11111110.

When a secondary station has a message to transmit and recognizes an EOP on the line, it changes the last binary 1 of the EOP to a 0 before transmission. This has the effect of turning the EOP into a flag pattern. The secondary station now places its message on the loop and terminates its message with an EOP. Any secondary stations further down the loop with messages to transmit can append their messages to the message of the first secondary station by the same process.

All secondary stations without messages to send merely echo the incoming messages and are prohibited from placing messages on the loop, except upon recognizing an EOP.

SDLC Loop mode is quite similar to normal SDLC mode except that two additional control bits are used. Writing a 1 to the Loop Mode bit in WR10 configures the SCC for Loop mode. Writing a 1 to the Go Active on Poll bit in the same register normally causes the SCC to change the next EOP into a flag and then begin transmitting on loop. However, when the SCC first goes on loop it uses the first EOP as a signal to insert the one-bit delay, and doesn't begin transmitting until it receives the second EOP. There are also two additional status bits in RR10, the On-Loop bit and the Loop-Sending bit.

There are also restrictions as to when and how a secondary station physically becomes part of the loop.

A secondary station that has just powered up must monitor the loop, without the one-bit-time delay, until it recognizes an EOP. When an EOP is recognized the one-bit-time delay is switched on. This does not disturb the loop because the line is marking idle between the time that the controller sends the EOP and the time that it receives the EOP back. The secondary station that has gone on-loop cannot place a message on the loop until the next time that an EOP is issued by the controller. A secondary station goes off loop in a similar manner. When given a command to go off-loop, the secondary station waits until the next EOP to remove the one-bit-time delay.

To operate the SCC in SDLC Loop mode, the SCC must first be programmed just as if normal SDLC were to be used. Loop mode is then selected by writing the appropriate control word in WR10.

The SCC is now waiting for the EOP so that it can go on loop. While waiting for the EOP, the SCC ties TxD to RxD with only the internal gate delays in the signal path. When the first EOP is recognized by the SCC, the Break/Abort/EOP bit is set in RR0, generating an External/Status interrupt (if so enabled). At the same time, the On-Loop bit in RR10 is set to indicate that the SCC is indeed on-loop, and a one-bit time delay is inserted in the TxD to the RxD path.

The SCC is now on-loop but cannot transmit a message until a flag and the next EOP are received. The requirement that a flag be received ensures that the SCC cannot erroneously send messages until the controller ends the current polling sequence and starts another one.

If the CPU in the secondary station with the SCC needs to transmit a message, the Go-Active-On-Poll bit in WR10 is set. If this bit is set when the EOP is detected, the SCC changes the EOP to a flag and starts sending another flag. The EOP is reported in the Break/Abort/EOP bit in RR0 and the CPU writes its data bytes to the SCC, just as in normal SDLC frame transmission. When the frame is complete and CRC has been sent, the SCC closes with a flag and reverts to One-Bit-Delay mode. The last zero of the flag, along with the marking line echoed from the RxD pin, form an EOP for secondary stations further down the loop.

While the SCC is actually transmitting a message, the loop-sending bit in R10 is set to indicate this.

If the Go-Active-On-Poll bit is not set at the time the EOP passes by, the SCC cannot send a message until a flag (terminating the current polling sequence) and another EOP are received.

If SDLC loop is deselected, the SCC is designed to exit from the loop gracefully. When the SDLC Loop mode is deselected by writing to WR10, the SCC waits until the next polling cycle to remove the one-bit time delay.

If a polling cycle is in progress at the time the command is written, the SCC finishes sending any message that it is transmitting, ends with an EOP, and disconnects TxD from RxD. If no message was in progress, the SCC immediately disconnects TxD from RxD.

Once the SCC is not sending on the loop, exiting from the loop is accomplished by setting the Loop Mode bit in WR10 to 0, and at the same time writing the Abort/Flag on Underrun and Mark/Flag idle bits with the desired values. The SCC will revert to normal SDLC operation as soon as an EOP is received, or immediately if the receiver is already in Hunt mode because of the receipt of an EOP.

To ensure proper loop operation after the SCC goes off the loop, and until the external relays take the SCC completely out of the loop, the SCC should be programmed for Mark idle instead of Flag idle. When the SCC goes off the loop, the On-Loop bit is reset.

**Note:** With NRZI encoding, removing the stations from the loop (removing the one-bit time delay) may cause problems further down the loop because of extraneous transitions on the line. The SCC avoids this problem by making transparent adjustments at the end of each frame it sends in response to an EOP. A response frame from the SCC is terminated by a flag and EOP. Normally, the flag and the EOP share a zero, but if such sharing would cause the RxD and TxD pins to be of opposite polarity after the EOP, the SCC adds another zero between the flag and the EOP. This causes an extra line transition so that RxD and TxD are identical after the EOP is sent. This extra zero is completely transparent because it only means that the flag and the EOP no longer share a zero. All that a proper loop exit needs, therefore, is the removal of the one-bit delay.

The SCC allows the user the option of using NRZI in SDLC Loop mode by programming WR10 appropriately. With NRZI encoding, the outputs of secondary stations in the loop are inverted from their inputs because of messages that they have transmitted.

Subsections 4.4.4.1 and 4.4.4.2 discuss the SDLC Loop Mode in Receive and Transmit.

#### 4.4.4.1 SDLC Loop Mode Receive

SDLC Loop mode is quite similar to SDLC mode except that two additional control bits are used. They are the Loop Mode bit (D1) and the Go-Active-On-Poll bit (D4) in WR10. In addition to these two extra control bits, there are also two status bits in RR10. They are the On Loop bit (D1) and the Loop Sending bit (D4).

Before Loop mode is selected, both the receiver and transmitter have to be completely initialized for SDLC operation. Once this is done, Loop mode is selected by setting bit D1 of WR10 to 1. At this point, the SCC connects TxD to RxD with only gate delays in the path. At the same time, a flag is loaded into the Transmit Shift register and is shifted to the end of the zero inserter, ready for transmission. The SCC remains in this state until the Go-Active-On-Poll bit (D4) in WR10 is set to 1. When this bit is set to 1, the receiver begins looking for a sequence of seven consecutive 1s, indicating either an EOP or an idle line. When the receiver detects this condition, the Break/Abort bit in RR0 is set to 1, and a one-bit time delay is inserted in the path from RxD to TxD.

The On-Loop bit in RR10 is also set to 1 at this time, and the receiver enters the Hunt mode. The SCC cannot transmit on the loop until a flag is received (causing the receiver to leave Hunt mode) and another EOP (bit pattern 11111110) is received. The SCC is now on the loop and capable of transmitting on the loop. As soon as this status is recognized by the processor, the Go-Active-On-Poll bit in WR10 is set to 0 to prevent the SCC from transmitting on the loop without a processor acknowledgment.

#### 4.4.4.2 SDLC Loop Mode Transmit

To transmit a message on the loop, the Go-Active-On-Poll bit in WR10 must be set to 1. Once this is done, the SCC changes the next received EOP into a Flag and begins transmitting on the loop.

When the EOP is received, the Break/Abort and Hunt bits in RR0 are set to 1, and the Loop Sending bit in RR10 is also set to 1. Data to be transmitted is written after the Go-Active-On-Poll bit has been set or after the receiver enters Hunt mode.

If the data is written immediately after the Go-Active-On-Poll bit has been set, the SCC only inserts one flag after the EOP is changed into a flag. If the data is not written until after the receiver enters the Hunt mode, the flags are transmitted until the data is written. If only one frame is to be transmitted on the loop in response to an EOP, the processor must set the Go Active on Poll bit to 0 before the last data is written to the transmitter. In this case, the transmitter closes the frame with a single flag and then reverts to the one-bit delay.

The Loop Sending bit in RR10 is set to 0 when the closing Flag has been sent. If more than one frame is to be transmitted, the Go-Active-On-Poll bit should not be set to 0 until the last frame is being sent. If this bit is not set to 0 before the end of a frame, the transmitter sends Flags until either more data is written to the transmitter, or until the Go-Active-On-Poll bit is set to 0. Note that the state of the Abort/Flag on Underrun and Mark/Flag idle bits in WR10 is ignored by the SCC in SDLC Loop mode.

## 4.3 BYTE-ORIENTED SYNCHRONOUS MODE (Continued)

#### 4.4.4.3 SDLC Loop Initialization

The initialization sequence for the SCC in SDLC Loop mode is similar to the sequence used in SDLC mode, except that it is longer. The processor should program WR4 first to select SDLC mode, and then WR10 to select the CRC preset value and program the Mark/Flag idle bit. The Loop Mode and Go-Active-On-Poll bits in WR10 should not be set to 1 yet. The flag is written in WR7 and the various options are selected in WR3 and WR5. At this point, the other registers are initialized as necessary (Table 4-12).

**Table 4-12. SDLC Loop Mode Initialization**

| | Bit Number | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Reg** | **D7** | **D6** | **D5** | **D4** | **D3** | **D2** | **D1** | **D0** | **Description** |
| WR4 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | Select x1 clock, SDLC mode, enable sync mode |
| WR3 | r | x | 0 | 1 | 1 | 1 | 0 | 0 | rx=# of Rx bits/char, No auto enable, enter Hunt, Enable Rx CRC, Address Search, No sync character load inhibit |
| WR5 | d | t | x | 0 | 0 | 0 | r | 1 | d=inverse of DTR pin, tx=# of Tx bits/char, use SDLC CRC, r=inverse state of /RTS pin, CRC enable |
| WR7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | SDLC Flag |
| WR6 | x | x | x | x | x | x | x | x | Receiver secondary address |
| WR15 | x | x | x | x | x | x | x | 1 | Enable access to new register |
| WR7' | 0 | 1 | 1 | d | 1 | r | 1 | 1 | Enable extended read, Tx INT on FIFO empty, d=REQUEST timing mode, Rx INT on 4 char, r=RTS deactivation, auto EOM reset, auto flag tx |
| WR10 | c | d | e | 1 | i | 0 | 1 | 0 | Enable Loop Mode, Go Active On Poll, c=CRC preset, de=data encoding method, i=idle line |
| WR3 | r | x | 0 | 1 | 1 | 1 | 0 | 1 | Enable Receiver |
| WR5 | d | t | x | 0 | 1 | 0 | r | 1 | Enable Transmitter |
| WR0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | Reset CRC generator |

The Loop Mode bit (D1) in WR10 is set to 1. When all of this is complete, the transmitter is enabled by setting bit D3 of WR5 to 1. Now that the transmitter is enabled, the CRC generator is initialized by issuing the Reset Tx CRC Generator command in WR0. The receiver is enabled by setting the Go-Active-On-Poll bit (D4) in WR10 to 1. The SCC goes on the loop when seven consecutive 1s are received, and signals this by setting the On-Loop bit in RR10. Note that the seven consecutive 1s will set the Break/Abort and Hunt bits in RR0 also. Once the SCC is on the loop, the Go-Active-On-Poll bit should be set to 0 until a message is to be transmitted on the loop. To transmit a message on the loop, the Go-Active-On-Poll bit should be set to 1. At this point, the processor may either write the first character to the transmit buffer and wait for a transmit buffer empty condition, or wait for the Break/Abort and Hunt bits to be set in RR10 and the Loop Sending bit to be set in RR10 before writing the first data to the transmitter. The Go-Active-On-Poll bit should be set to 0 after the transition of the frame has begun. To go off of the loop, the processor should set the Go-Active-On-Poll bit in WR10 to 0 and then wait for the Loop Sending bit in RR10 to be set to 0. At this point, the Loop Mode bit (D1) in WR10 is set to 0 to request an orderly exit from the loop. The SCC exits SDLC Loop mode when seven consecutive 1s have been received; at the same time the Break/Abort and Hunt bits in RR0 are set to 1, and the On Loop bit in RR10 is set to 0.

# CHAPTER 5
## REGISTER DESCRIPTIONS

## 5.1 INTRODUCTION

This section describes the functions of the various bits in the registers of the SCC (Tables 5-1 and 5-2). Reserved bits are not used in this implementation of the device and may or may not be physically present in the device. For the register addresses, also refer to Tables 2-1, 2-2 and 2-5 in Chapter 2. Reserved bits that are physically present are readable and writable but reserved bits that are not present will always be read as zero. To ensure compatibility with future versions of the device, reserved bits should always be written with zeros. Reserved commands are not used for the same reason.

. **Table 5-1. SCC Write Registers**

| Reg | Description |
|---|---|
| WR0 | Reg. pointers, various initialization commands |
| WR1 | Transmit and Receive interrupt enables, WAIT/DMA commands |
| WR2 | Interrupt Vector |
| WR3[2] | Receive parameters and control modes |
| WR4[2] | Transmit and Receive modes and parameters |
| WR5[2] | Transmit parameters and control modes |
| WR6 | Sync Character or SDLC address |
| WR7 | Sync Character or SDLC flag |
| WR7'[1] | Extended Feature and FIFO Control (WR7 Prime) |
| WR8 | Transmit buffer |
| WR9 | Master Interrupt control and reset commands |
| WR10[2] | Miscellaneous transmit and receive control bits |
| WR11 | Clock mode controls for receive and transmit |
| WR12 | Lower byte of baud rate generator |
| WR13 | Upper byte of baud rate generator |
| WR14 | Miscellaneous control bits |
| WR15 | External status interrupt enable control |

**Table 5-2. SCC Read Registers**

| Reg | Description |
|---|---|
| RR0 | Transmit and Receive buffer status and external status |
| RR1 | Special Receive Condition status |
| RR2 | Modified interrupt vector (Channel B only), Unmodified interrupt vector (Channel A only) |
| RR3 | Interrupt pending bits (Channel A only) |
| RR4[2] | Transmit and Receive modes and parameters (WR4) |
| RR5[2] | Transmit parameters and control modes (WR5) |
| RR6[3] | SDLC FIFO byte counter lower byte (only when enabled) |
| RR7[3] | SDLC FIFO byte count and status (only when enabled) |
| RR8 | Receive buffer |
| RR9[2] | Receive parameters and control modes (WR3) |
| RR10 | Miscellaneous status bits |
| RR11[2] | Miscellaneous transmit and receive control bits (WR10) |
| RR12 | Lower byte of baud rate generator time constant |
| RR13 | Upper byte of baud rate generator time constant |
| RR14[2] | Extended Feature and FIFO Control (WR7 Prime) |
| RR15 | External Status interrupt information |

**Notes for Tables 5-1 and 5-2:**

1. ESCC and 85C30 only.
2. On the ESCC and 85C30, these registers are readable as RR9, RR4, RR5, and RR11, respectively, when WR7' D6=1. Refer to the description of WR7 Prime for enabling the extended read capability.
3. This feature is not available on NMOS.

## 5.1 INTRODUCTION (Continued)

Among these registers, WR9 (Master Interrupt Control and Reset register) can be accessed through either channel. The RR2 (Interrupt Vector register) returns the interrupt vector modified by status, if read from Channel B, and written value (without modification), if read from Channel A.

Channel A has an additional read register which contains all the Interrupt Pending bits (RR3A).

**Write Registers.** Eleven write registers are used for control (includes transmit buffer/FIFO); two for sync character generation/detection; two for baud rate generation. In addition, there are two write registers which are shared by both channels; one is the interrupt vector register (WR2); the other is the Master Interrupt and Reset register (WR9).

On the ESCC and 85C30, there is one additional register (WR7') to control enhanced features.

See Table 5-1 for a summary of Write registers.

**Read Registers.** Four read registers indicate status information; two are for baud rate generation; one for the receive buffer. In addition, there are two read registers which are shared by both channels; one for the interrupt pending bits; another for the interrupt vector. On the CMOS/ESCC, there are two additional registers, RR6 and RR7. They are available if the Frame Status FIFO feature was enabled in the SDLC mode of operation. On the ESCC, there is an "extended read" option and if its enabled, certain write registers can be read back.

See Table 5-2 for a summary of Read registers.

## 5.2 WRITE REGISTERS

The SCC write register set in each channel has 11 control registers (includes transmit buffer/FIFO), two sync character registers, and two baud rate time constant registers. The interrupt control register and the master interrupt control and reset register are shared by both channels. In addition to these, the ESCC and 85C30 has a register (WR7'; prime 7) to control the enhancements.

Between 80X30 and 85X30, the variation in register definition is a command decode structure; Write Register 0 (WR0). The following sections describe in detail each write register and the associated bit configuration for each.

The following sections describe WR registers in detail:

### 5.2.1 Write Register 0 (Command Register)

WR0 is the command register and the CRC reset code register. WR0 takes on slightly different forms depending upon whether the SCC is in the Z85X30 or the Z80X30. Figure 5-1 shows the bit configuration for the Z85X30 and includes register select bits in addition to command and reset codes.

Figure 5-2 shows the bit configuration for the Z80X30 and includes (in Channel B only) the address decoding select described later.

The following bit description for WR0 is identical for both versions except where specified:

**Bits D7 and D6: CRC Reset Codes 1 And 0.**

**Null Command (00).** This command has no effect on the SCC and is used when a write to WR0 is necessary for some reason other than a CRC Reset command.

**Reset Receive CRC Checker Command (01).** This command is used to initialize the receive CRC circuitry. It is necessary in synchronous modes (except SDLC) if the Enter Hunt Mode command in Write Register 3 is not issued between received messages. Any action that disables the receiver initializes the CRC circuitry. Resetting the Receive CRC Checker command is accomplished automatically in SDLC mode.

**Reset Transmit CRC Generator Command (10).** This command initializes the CRC generator. It is usually issued in the initialization routine and after the CRC has been transmitted. A Channel Reset does not initialize the generator and this command is not issued until after the transmitter has been enabled in the initialization routine.

On the ESCC and 85C30, this command is not needed if Auto EOM Reset mode is enabled (WR7' D1=1).

**Reset Transmit Underrun/EOM Latch Command (11).** This command controls the transmission of CRC at the end of transmission (EOM). If this latch has been reset, and a transmit underrun occurs, the SCC automatically appends CRC to the message. In SDLC mode with Abort on Underrun selected, the SCC sends an abort and Flag on underrun if the TX Underrun/EOM latch has been reset.

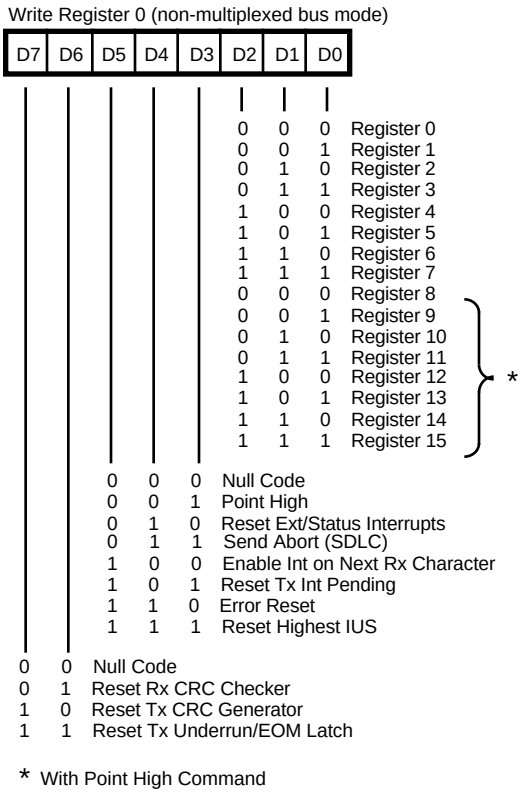



**Figure 5-1. Write Register 0 in the Z85X30**

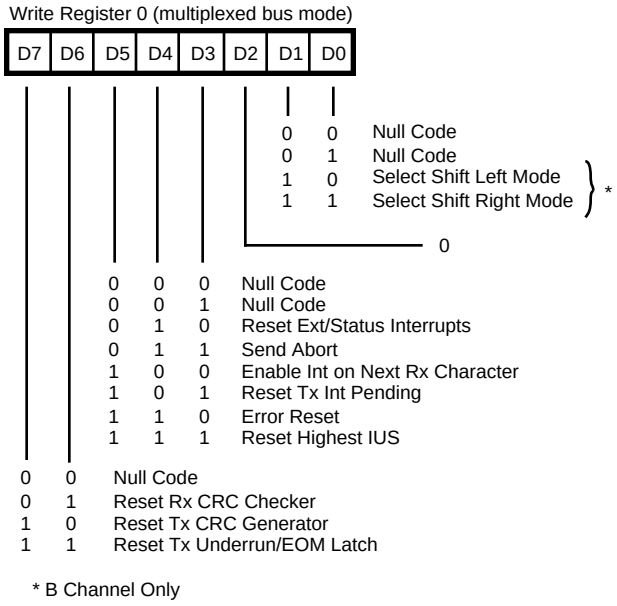



**Figure 5-2. Write Register 0 in the Z80X30**

At the start of the CRC transmission, the Tx Underrun/EOM latch is set. The Reset command can be issued at any time during a message. If the transmitter is disabled, this command does not reset the latch. However, if no External Status interrupt is pending, or if a Reset External Status interrupt command accompanies this command while the transmitter is disabled, an External/Status interrupt is generated with the Tx Underrun/EOM bit reset in RR0.

**Bits D5-D3: Command Codes for the SCC.**

**Null Command (000).** The Null command has no effect on the SCC.

**Point High Command (001).** This command effectively adds eight to the Register Pointer (D2-D0) by allowing WR8 through WR15 to be accessed. The Point High command and the Register Pointer bits are written simultaneously. This command is used in the Z85X30 version of the SCC. Note that WR0 changes form depending upon the SCC version. Register access for the Z80X30 version of the SCC is accomplished through direct addressing.

**Reset External/Status Interrupts Command (010).** After an External/Status interrupt (a change on a modem line or a break condition, for example), the status bits in RR0 are latched. This command re-enables the bits and allows interrupts to occur again as a result of a status change. Latching the status bits captures short pulses until the CPU has time to read the change.

The SCC contains simple queueing logic associated with most of the external status bits in RR0. If another External/Status condition changes while a previous condition is still pending (Reset External/Status Interrupt has not yet been issued) and this condition persists until after the command is issued, this second change causes another External/Status interrupt. However, if this second status change does not persist (there are two transitions), another interrupt is not generated. Exceptions to this rule are detailed in the RR0 description.

**Send Abort Command (011).** This command is used in SDLC mode to transmit a sequence of eight to thirteen 1s. This command always empties the transmit buffer and sets Tx Underrun/EOM bit in Read Register 0.

**Enable Interrupt On Next Rx Character Command (100).** If the interrupt on First Received Character mode is selected, this command is used to reactivate that mode after each message is received. The next character to enter the Receive FIFO causes a Receive interrupt. Alternatively, the first previously stored character in the FIFO causes a Receive interrupt.

## 5.1 INTRODUCTION (Continued)

**Reset Tx Interrupt Pending Command (101).** This command is used in cases where there are no more characters to be sent; e.g., at the end of a message. This command prevents further transmit interrupts until after the next character has been loaded into the transmit buffer or until CRC has been completely sent. This command is necessary to prevent the transmitter from requesting an interrupt when the transmit buffer becomes empty (with Transmit Interrupt Enabled).

**Error Reset Command (110).** This command resets the error bits in RR1. If interrupt on first Rx Character or Interrupt on Special Condition modes is selected and a special condition exists, the data with the special condition is held in the Receive FIFO until this command is issued. If either of these modes is selected and this command is issued before the data has been read from the Receive FIFO, the data is lost.

**Reset Highest IUS Command (110).** This command resets the highest priority Interrupt Under Service (IUS) bit, allowing lower priority conditions to request interrupts. This command allows the use of the internal daisy chain (even in systems without an external daisy chain) and is the last operation in an interrupt service routine.

**Bits 2 through 0: Register Selection Code**

On the Z85X30, these three bits select Registers 0 through 7. With the Point High command, Registers 8 through 15 are selected (Table 5-3).

In the multiplexed bus mode, bits D2 through D0 have the following function.

Bit D2 must be programmed as 0. Bits D1 and D0 select Shift Left/Right; that is WR0 (1-0)=10 for shift left and WR0 (1-0)=11 for shift right. See Section 2.1.4 for further details on Z80X30 register access.

### 5.2.2 Write Register 1 (Transmit/Receive Interrupt and Data Transfer Mode Definition)

Write Register 1 is the control register for the various SCC interrupt and Wait/Request modes. Figure 5-3 shows the bit assignments for WR1.

**Bit 7: WAIT/DMA Request Enable.**
This bit enables the Wait/Request function in conjunction with the Request/Wait Function Select bit (D6).

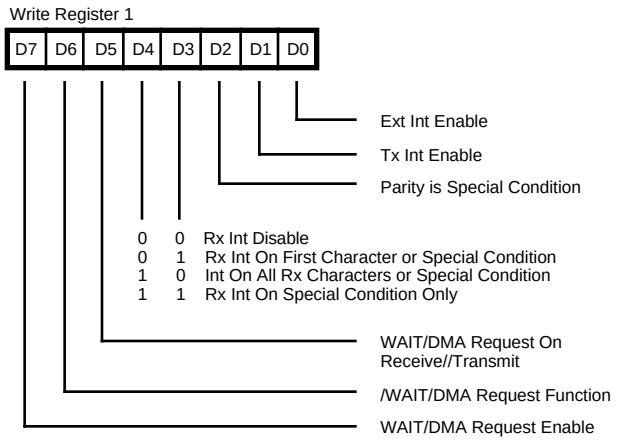



**Figure 5-3. Write Register 1**

When programmed to 0, the selected function (bit 6) forces the /W//REQ pin into the appropriate inactive state (High for Request, floating for Wait).

When programmed to 1, the state of bit 6 determines the activity of the /W//REQ pin (Wait or Request).

**Bit 6: WAIT/DMA Request Function**
When programmed to 0, the Wait function is selected. In the Wait mode, the /W//REQ pin switches from floating to Low when the CPU attempts to transfer data before the SCC is ready.

When programmed to 1, the Request function is selected. In the Request mode, the /W//REQ pin switches from High to Low when the SCC is ready to transfer data.

**Bit 5: /WAIT//REQUEST on Transmit or Receive**
When programmed to 0, the state of the /W//REQ pin is determined by bit 6 and the state of the transmit buffer.

**Note:** A transmit request function is available on the /DTR//REQ pin. This allows full-duplex operation under DMA control for both channels.

**Table 5-3. Z85X30 Register Map**

| A//B | PNT2 | PNT1 | PNT0 | WRITE | READ 8530 85C30/230* WR15 D2 = 0 | 85C30/230 WR15 D2=1 | 85C30/85230W R15 D2=1 WR7' D6=1 |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | WR0B | RR0B | RR0B | RR0B |
| 0 | 0 | 0 | 1 | WR1B | RR1B | RR1B | RR1B |
| 0 | 0 | 1 | 0 | WR2 | RR2B | RR2B | RR2B |
| 0 | 0 | 1 | 1 | WR3B | RR3B | RR3B | RR3B |
| 0 | 1 | 0 | 0 | WR4B | (RR0B) | (RR0B) | (WR4B) |
| 0 | 1 | 0 | 1 | WR5B | (RR1B) | (RR1B) | (WR5B) |
| 0 | 1 | 1 | 0 | WR6B | (RR2B) | RR6B | RR6B |
| 0 | 1 | 1 | 1 | WR7B | (RR3B) | RR7B | RR7B |
| 1 | 0 | 0 | 0 | WR0A | RR0A | RR0A | RR0A |
| 1 | 0 | 0 | 1 | WR1A | RR1A | RR1A | RR1A |
| 1 | 0 | 1 | 0 | WR2 | RR2A | RR2A | RR2A |
| 1 | 0 | 1 | 1 | WR3A | RR3A | RR3A | RR3A |
| 1 | 1 | 0 | 0 | WR4A | (RR0A) | (RR0A) | (WR4A) |
| 1 | 1 | 0 | 1 | WR5A | (RR1A) | (RR1A) | (WR5A) |
| 1 | 1 | 1 | 0 | WR6A | (RR2A) | RR6A | RR6A |
| 1 | 1 | 1 | 1 | WR7A | (RR3A) | RR7A | RR7A |
| **With Point High Command** | | | | | | | |
| 0 | 0 | 0 | 0 | WR8B | RR8B | RR8B | RR8B |
| 0 | 0 | 0 | 1 | WR9 | (RR13B) | (RR13B) | (WR3B) |
| 0 | 0 | 1 | 0 | WR10B | RR10B | RR10B | RR10B |
| 0 | 0 | 1 | 1 | WR11B | (RR15B) | (RR15B) | (WR10B) |
| 0 | 1 | 0 | 0 | WR12B | RR12B | RR12B | RR12B |
| 0 | 1 | 0 | 1 | WR13B | RR13B | RR13B | RR13B |
| 0 | 1 | 1 | 0 | WR14B | RR14B | RR14B | (WR7'B) |
| 0 | 1 | 1 | 1 | WR15B | RR15B | RR15B | RR15B |
| 1 | 0 | 0 | 0 | WR8A | RR8A | RR8A | RR8A |
| 1 | 0 | 0 | 1 | WR9A | (RR13A) | (RR13A) | (WR3A) |
| 1 | 0 | 1 | 0 | WR10A | RR10A | RR10A | RR10A |
| 1 | 0 | 1 | 1 | WR11A | (RR15A) | (RR15A) | (WR10A) |
| 1 | 1 | 0 | 0 | WR12A | RR12A | RR12A | RR12A |
| 1 | 1 | 0 | 1 | WR13A | RR13A | RR13A | RR13A |
| 1 | 1 | 1 | 0 | WR14A | RR14A | RR14A | (WR7'A) |
| 1 | 1 | 1 | 1 | WR15A | RR15A | RR15A | RR15A |

**Notes:**
WR15 bit D2 enables status FIFO function. (Not available on NMOS)
WR7' bit D6 enables extend read function. (Only on ESCC and 85C30)
* Includes 85C30 and 85230 with WR15 D2=0.

## 5.1 INTRODUCTION (Continued)

When programmed to 1, this bit allows the Wait/Request function to follow the state of the receive buffer. Thus, depending on the state of bit 6, the /W//REQ pin is active or inactive in relation to the empty or full state of the receive buffer.

The request function occurs only when the SCC is not selected; e.g., if the internal request becomes active while the SCC is in the middle of a read or write cycle, the external request does not become active until the cycle is complete. An active request output causes a DMA controller to initiate a read or write operation. If the request on Transmit mode is selected in either SDLC or Synchronous Mode, the Request pin is pulsed Low for one PCLK cycle at the end of CRC transmission to allow the immediate transmission of another block of data.

In the Wait On Receive mode, the /WAIT pin is active if the CPU attempts to read SCC data that has not yet been received. In the Wait On Transmit mode, the /WAIT pin is active if the CPU attempts to write data when the transmit buffer is still full. Both situations occur frequently when block transfer instructions are used.

**Bits 4 and 3: Receive Interrupt Modes**
**Receive Interrupts Disabled (00).** This mode prevents the receiver from requesting an interrupt. It is normally used in a polled environment where either the status bits in RR0 or the modified vector in RR2 (Channel B) are monitored to initiate a service routine. Although the receiver interrupts are disabled, a special condition can still provide a unique vector status in RR2.

**Receive Interrupt on First Character or Special Condition (01).** The receiver requests an interrupt in this mode on the first available character (or stored FIFO character) or on a special condition. Sync characters, stripped from the message stream, do not cause interrupts.

Special receive conditions are: receiver overrun, framing error, end of frame, or parity error (if selected). If a special receive condition occurs, the data containing the error is stored in the Receive FIFO until an Error Reset command is issued by the CPU.

This mode is usually selected when a Block Transfer mode is used. In this interrupt mode, a pending special receive condition remains set until either an error Reset command, a channel or hardware reset, or until receive interrupts are disabled.

The Receive Interrupt on First Character or Special Condition mode can be re-enabled by the Enable Rx Interrupt on Next Character command in WR0.

***ESCC:***
***See the description of WR7' on how this function can be changed.***

**Interrupt on All Receive Characters or Special Condition (10).** This mode allows an interrupt for every character received (or character in the Receive FIFO) and provides a unique vector when a special condition exists. The Receiver Overrun bit and the Parity Error bit in RR1 are two special conditions that are latched. These two bits are reset by the Error Reset command. Receiver overrun is always a special receive condition, and parity can be programmed to be a special condition.

Data characters with special receive conditions are not held in the Receive FIFO in the Interrupt On All Receive Characters or Special Conditions Mode as they are in the other receive interrupt modes.

**Receive Interrupt on Special Condition (11).** This mode allows the receiver to interrupt only on characters with a special receive condition. When an interrupt occurs, the data containing the error is held in the Receive FIFO until an Error Reset command is issued. When using this mode in conjunction with a DMA, the DMA is initialized and enabled before any characters have been received by the ESCC. This eliminates the time-critical section of code required in the Receive Interrupt on First Character or Special Condition mode. Hence, all data can be transferred via the DMA so that the CPU need not handle the first received character as a special case. In SDLC mode, if the SDLC Frame Status FIFO is enabled and an EOF is received, an interrupt with vector for receive data available is generated and the Receive FIFO is not locked.

**Bit 2: Parity Is Special Condition**
If this bit is set to 1, any received characters with parity not matching the sense programmed in WR4 give rise to a Special Receive Condition. If parity is disabled (WR4), this bit is ignored. A special condition modifies the status of the interrupt vector stored in WR2. During an interrupt acknowledge cycle, this vector can be placed on the data bus.

**Bit 1: Transmitter Interrupt Enable**
If this bit is set to 1, the transmitter requests an interrupt whenever the transmit buffer becomes empty.

**Bit 0: External/Status Master Interrupt Enable**
This bit is the master enable for External/Status interrupts including /DCD, /CTS, /SYNC pins, break, abort, the beginning of CRC transmission when the Transmit/Underrun/EOM latch is set, or when the counter in the baud rate generator reaches 0. Write Register 15 contains the individual enable bits for each of these sources of External/Status interrupts. This bit is reset by a channel or hardware reset.

### 5.2.3 Write Register 2 (Interrupt Vector)

WR2 is the interrupt vector register. Only one vector register exists in the SCC, and it can be accessed through either channel. The interrupt vector can be modified by status information. This is controlled by the Vector Includes Status (VIS) and the Status High/Status Low bits in WR9. The bit positions for WR2 are shown in Figure 5-4.

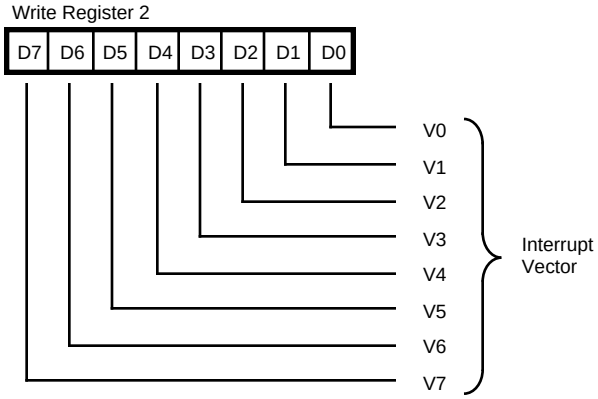



**Figure 5-4. Write Register 2**

### 5.2.4 Write Register 3 (Receive Parameters and Control)

This register contains the control bits and parameters for the receiver logic as illustrated in Figure 5-5. On the ESCC and 85C30, with the Extended Read option enabled, this register may be read as RR9.

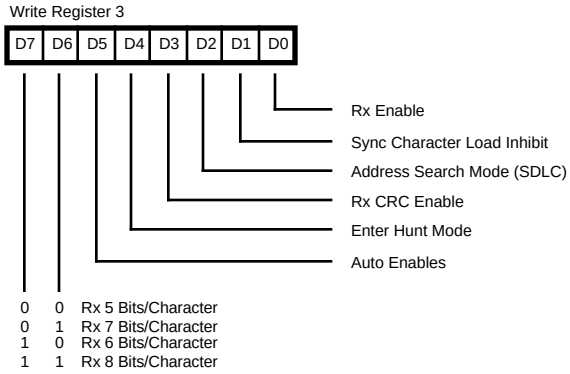



**Figure 5-5. Write Register 3**

**Bits 7 and 6: Receiver Bits/Character**
The state of these two bits determines the number of bits to be assembled as a character in the received serial data stream. The number of bits per character can be changed while a character is being assembled, but only before the number of bits currently programmed is reached. Unused bits in the Received Data Register (RR8) are set to 1 in asynchronous modes. In Synchronous and SDLC modes, the SCC merely transfers an 8-bit section of the serial data stream to the Receive FIFO at the appropriate time. Table 5-4 lists the number of bits per character in the assembled character format.

**Table 5-4. Receive Bits per Character**

| D7 | D6 | Bits/Character |
|---|---|---|
| 0 | 0 | 5 |
| 0 | 1 | 7 |
| 1 | 0 | 6 |
| 1 | 1 | 8 |

**Bit 5: Auto Enable**
This bit programs the function for both the /DCD and /CTS pins. /CTS becomes the transmitter enable and /DCD becomes the receiver enable when this bit is set to 1. However, the Receiver Enable and Transmit Enable bits must be set before the /DCD and /CTS pins can be used in this manner. When the Auto Enable bit is set to 0, the /DCD and /CTS pins are inputs to the corresponding status bits in Read Register 0. The state of /DCD is ignored in the Local Loopback mode. The state of /CTS is ignored in both Auto Echo and Local Loopback modes.

**Bit 4: Enter Hunt Mode**
This command forces the comparison of sync characters or flags to assembled receive characters for the purpose of synchronization. After reset, the SCC automatically enters the Hunt mode (except asynchronous). Whenever a flag or sync character is matched, the Sync/Hunt bit in Read Register 0 is reset and, if External/Status Interrupt Enable is set, an interrupt sequence is initiated. The SCC automatically enters the Hunt mode when an abort condition is received or when the receiver is enabled.

**Bit 3: Receiver CRC Enable**
This bit is used to initiate CRC calculation at the beginning of the last byte transferred from the Receiver Shift register to the Receive FIFO. This operation occurs independently of the number of bytes in the Receive FIFO. When a particular byte is to be excluded from the CRC calculation, this bit should be reset before the next byte is transferred to the Receive FIFO. If this feature is used, care must be taken to ensure that eight bits per character is selected in the receiver because of an inherent delay from the Receive Shift register to the CRC checker.

## 5.1 INTRODUCTION (Continued)

This bit is internally set to 1 in SDLC mode and the SCC calculates the CRC on all bits except zeros inserted between the opening and closing flags. This bit is ignored in asynchronous modes.

**Bit 2: Address Search Mode (SDLC)**
Setting this bit in SDLC mode causes messages with addresses not matching the address programmed in WR6 to be rejected. No receiver interrupts occur in this mode unless there is an address match. The address that the SCC attempts to match is unique (1 in 256) or multiple (16 in 256), depending on the state of Sync Character Load Inhibit bit. Address FFH is always recognized as a global address. The Address Search mode bit is ignored in all modes except SDLC.

**Bit 1: SYNC Character Load Inhibit**
If this bit is set to 1 in any mode except SDLC, the SCC compares the byte in WR6 with the byte about to be stored in the FIFO, and it inhibits this load if the bytes are equal. (Caution: this also occurs in the asynchronous mode if the received character matches the contents of WR6.) The SCC does not calculate the CRC on bytes stripped from the data stream in this manner. If the 6-bit sync option is selected while in Monosync mode, the comparison is still across eight bits, so WR6 is programmed for proper operation.

If the 6-bit sync option is selected with this bit set to 1, all sync characters except the one immediately preceding the data are stripped from the message. If the 6-bit sync option is selected while in the Bisync mode, this bit is ignored.

The address recognition logic of the receiver is modified in SDLC mode if this bit is set to 1, i.e., only the four most significant bits of WR6 must match the receiver address. This procedure allows the SCC to receive frames from up to 16 separate sources without programming WR6 for each source (if each station address has the four most significant bits in common). The address field in the frame is still eight bits long. Address FFH is always recognized as a global address.

The bit is ignored in SDLC mode if Address Search mode has not been selected.

**Bit 0: Receiver Enable**
When this bit is set to 1, receiver operation begins. This bit should be set only after all other receiver parameters are established and the receiver is completely initialized. This bit is reset by a channel or hardware reset command, and it disables the receiver.

### 5.2.5 Write Register 4 (Transmit/Receive Miscellaneous Parameters and Modes)

WR4 contains the control bits for both the receiver and the transmitter. These bits should be set in the transmit and receiver initialization routine before issuing the contents of WR1, WR3, WR6, and WR7. Bit positions for WR4 are shown in Figure 5-6. On the ESCC and 85C30, with the Extended Read option enabled, this register is read as RR4.

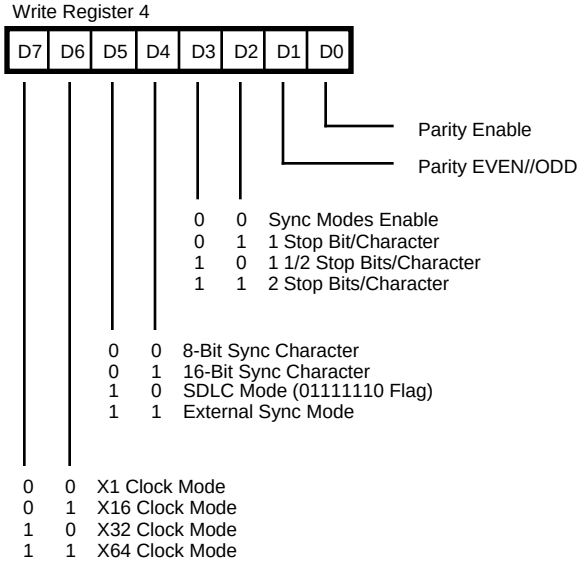



**Figure 5-6. Write Register 4**

**Bits 7 and 6: Clock Rate bits 1 and 0**
These bits specify the multiplier between the clock and data rates. In synchronous modes, the 1X mode is forced internally and these bits are ignored unless External Sync mode has been selected.

**1X Mode (00).** The clock rate and data rate are the same. In External Sync mode, this bit combination specifies that only the /SYNC pin is used to achieve character synchronization.

**16X Mode (01).** The clock rate is 16 times the data rate. In External Sync mode, this bit combination specifies that only the /SYNC pin is used to achieve character synchronization.

**32X Mode (10).** The clock rate is 32 times the data rate. In External Sync mode, this bit combination specifies that either the /SYNC pin or a match with the character stored in WR7 will signal character synchronization. The sync character can be either six or eight bits long as specified by the 6-bit/8-bit sync bit in WR10.

**64X Mode (11).** The clock rate is 64 times the data rate. With this bit combination in External Sync mode, both the receiver and transmitter are placed in SDLC mode. The only variation from normal SDLC operation is that the /SYNC pin is used to start or stop the reception of a frame by forcing the receiver to act as though a flag had been received.

**Bits 5 and 4: SYNC Mode selection bits 1 and 0**
These two bits select the various options for character synchronization. They are ignored unless synchronous modes are selected in the stop bits field of this register.

**Monosync Mode (00).** In this mode, the receiver achieves character synchronization by matching the character stored in WR7 with an identical character in the received data stream. The transmitter uses the character stored in WR6 as a time fill. The sync character is either six or eight bits, depending on the state of the 6-bit/8-bit sync bit in WR10. If the Sync Character Load Inhibit bit is set, the receiver strips the contents of WR6 from the data stream if received within character boundaries.

**Bisync Mode (01).** The concatenation of WR7 with WR6 is used for receiver synchronization and as a time fill by the transmitter. The sync character is 12 or 16 bits in the receiver, depending on the state of the 6-bit/8-bit sync bit in WR10. The transmitted character is always 16 bits.

**SDLC Mode (10).** In this mode, SDLC is selected and requires a Flag (01111110) to be written to WR7. The receiver address field is written to WR6. The SDLC CRC polynomial is also selected (WR5) in SDLC mode.

**External Sync Mode (11).** In this mode, the SCC expects external logic to signal character synchronization via the /SYNC pin. If the crystal oscillator option is selected (in WR11), the internal /SYNC signal is forced to 0. In this mode, the transmitter is in Monosync mode using the contents of WR6 as the time fill with the sync character length specified by the 6-bit/8-bit Sync bit in WR10.

**Bits 3 and 2: Stop Bits selection, bits 1 and 0**
These bits determine the number of stop bits added to each asynchronous character that is transmitted. The receiver always checks for one stop bit in Asynchronous mode. A special mode specifies that a Synchronous mode is to be selected. D2 is always set to 1 by a channel or hardware reset to ensure that the /SYNC pin is in a known state after a reset.

**Synchronous Modes Enable (00).** This bit combination selects one of the synchronous modes specified by bits D4, D5, D6, and D7 of this register and forces the 1X Clock mode internally.

**1 Stop Bit/Character (01).** This bit selects Asynchronous mode with one stop bit per character.

**1 1/2 Stop Bits/Character (10).** These bits select Asynchronous mode with 1-1/2 stop bits per character. This mode is not used with the 1X clock mode.

**2 Stop Bits/Character (11).** These bits select Asynchronous mode with two stop bits per transmitted character and checks for one received stop bit.

**Bit 1: Parity Even//Odd select bit**
This bit determines whether parity is checked as even or odd. A 1 programmed here selects even parity, and a 0 selects odd parity. This bit is ignored if the Parity enable bit is not set.

**Bit 0: Parity Enable**
When this bit is set, an additional bit position beyond those specified in the bits/character control is added to the transmitted data and is expected in the receive data. The Received Parity bit is transferred to the CPU as part of the data unless eight bits per character is selected in the receiver.

## 5.2.6 Write Register 5 (Transmit Parameters and Controls)

WR5 contains control bits that affect the operation of the transmitter. D2 affects both the transmitter and the receiver. Bit positions for WR5 are shown in Figure 5-7. On the 85X30 with the Extended Read option enabled, this register is read as RR5.

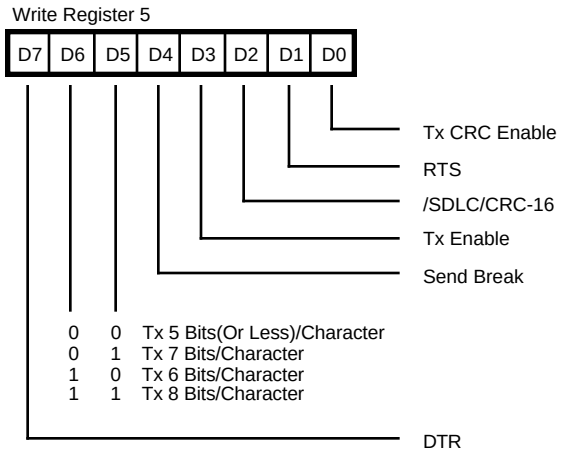



**Figure 5-7. Write Register 5**

**Bit 7: Data Terminal Ready control bit**
This is the control bit for the /DTR//REQ pin while the pin is in the DTR mode (selected in WR14). When set, /DTR is Low; when reset, /DTR is High. This bit is ignored when /DTR//REQ is programmed to act as a /REQ pin. This bit is reset by a channel or hardware reset.

## 5.1 INTRODUCTION (Continued)

**Bits 6 and 5: Transmit Bits/Character select bits 1 and 0**
These bits control the number of bits in each byte transferred to the transmit buffer. Bits sent must be right justified with the least significant bits first.

The Five Or Less mode allows transmission of one to five bits per character. For five or fewer bits per character, the data character must be formatted as shown below in Table 5-5. In the Six or Seven Bits/Character modes, unused data bits are ignored.

**Bit 4: Send Break control bit**
When set, this bit forces the TxD output to send continuous 0s beginning with the following transmit clock, regardless of any data being transmitted at the time. This bit functions whether or not the transmitter is enabled. When reset, TxD continues to send the contents of the Transmit Shift register, which might be syncs, data, or all 1s. If this bit is set while in the X21 mode (Monosync and Loop mode selected) and character synchronization is achieved in the receiver, this bit is automatically reset and the transmitter begins sending syncs or data. This bit is also reset by a channel or hardware reset.

**Table 5-5. Transmit Bits per Character**

| Bit 7 | Bit 6 | |
|---|---|---|
| 0 | 0 | 5 or less bits/character |
| 0 | 1 | 7 bits/character |
| 1 | 0 | 6 bits/character |
| 1 | 1 | 8 bits/character |

**Note:** For five or less bits per character selection in WR5, the following encoding is used in the data sent to the transmitter. D is the data bit(s) to be sent.

| D7 | D6 | D5 | D4 | D3 | D2 | D1 | D0 | |
|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | D | Sends one data bit |
| 1 | 1 | 1 | 0 | 0 | 0 | D | D | Sends two data bits |
| 1 | 1 | 0 | 0 | 0 | D | D | D | Sends three data bits |
| 1 | 0 | 0 | 0 | D | D | D | D | Sends four data bits |
| 0 | 0 | 0 | D | D | D | D | D | Sends five data bits |

**Bit 3: Transmit Enable**
Data is not transmitted until this bit is set, and the TxD output sends continuous 1s unless Auto Echo mode or SDLC Loop mode is selected. If this bit is reset after transmission starts, the transmission of data or sync characters is completed. If the transmitter is disabled during the transmission of a CRC character, sync or flag characters are sent instead of CRC. This bit is reset by a channel or hardware reset.

**Bit 2: SDLC/CRC-16 polynomial select bit**
This bit selects the CRC polynomial used by both the transmitter and receiver. When set, the CRC-16 polynomial is used; when reset, the SDLC polynomial is used. The SDLC/CRC polynomial is selected when SDLC mode is selected. The CRC generator and checker can be preset to all 0s or all 1s, depending on the state of the Preset 1/Preset 0 bit in WR10.

**Bit 1: Request To Send control bit**
This is the control bit for the /RTS pin. When the RTS bit is set, the /RTS pin goes Low; when reset, /RTS goes High. When Auto Enable is set in asynchronous mode, the /RTS pin immediately goes Low when the RTS bit is set. However, when the RTS bit is reset, the /RTS pin remains Low until the transmitter is completely empty and the last stop bit has left the TxD pin. In synchronous modes, the /RTS pin directly follows the state of this bit, except in SDLC mode under specific conditions. In SDLC mode, if Flag On Underrun bit (WR10, D2) is set, RTS bit in WR5 is reset, and D2 in WR7' is set. The /RTS pin deasserts automatically at the last bit of the closing flag triggered by the rising edge of the Tx clock. This bit is reset by a channel or hardware reset.

**Bit 0: Transmit CRC Enable**
This bit determines whether or not the CRC is calculated on a transmit character. If this bit is set at the time the character is loaded from the transmit buffer to the Transmit Shift register, the CRC is calculated on that character. The CRC is not automatically sent unless this bit is set when the transmit underrun exists.

### 5.2.7 Write Register 6 (Sync Characters or SDLC Address Field)

WR6 is programmed to contain the transmit sync character in the Monosync mode, or the first byte of a 16-bit sync character in the External Sync mode. WR6 is not used in asynchronous modes. In the SDLC modes, it is programmed to contain the secondary address field used to compare against the address field of the SDLC Frame. In SDLC mode, the SCC does not automatically transmit the station address at the beginning of a response frame. Bit positions for WR6 are shown in Figure 5-8.

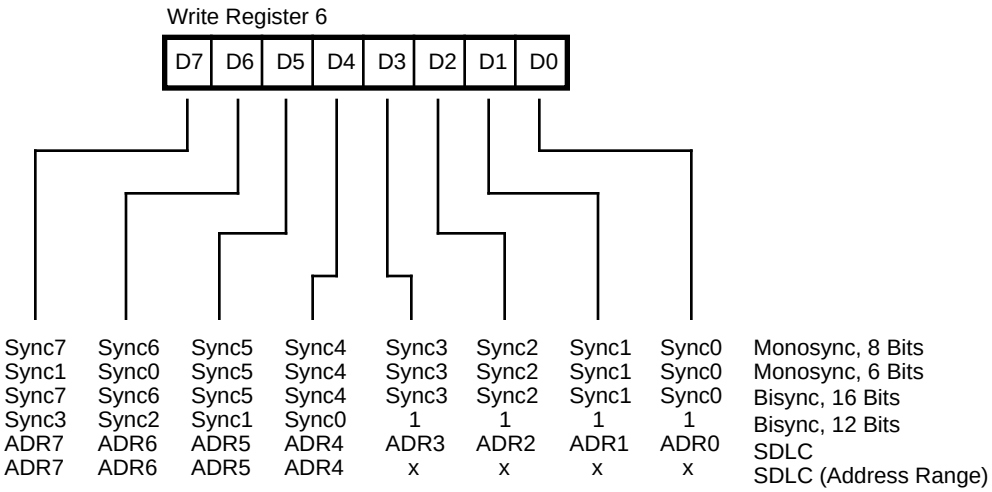



**Figure 5-8. Write Register 6**

## 5.2.8 Write Register 7 (Sync Character or SDLC Flag)

WR7 is programmed to contain the receive sync character in the Monosync mode, a second byte (the last eight bits) of a 16-bit sync character in the Bisync mode, or a Flag character (01111110) in the SDLC modes. WR7 holds the receive sync character or a flag if one of the special versions of the External Sync mode is selected. WR7 is not used in Asynchronous mode. Bit positions for WR7 are shown in Figure 5-9.

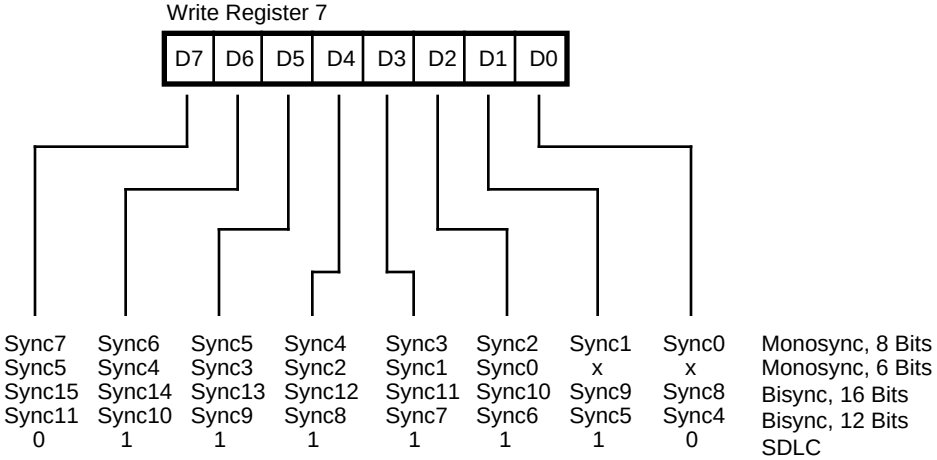



**Figure 5-9. Write Register 7**

## 5.1 INTRODUCTION (Continued)

### 5.2.9 Write Register 7 Prime (ESCC only)

This Register is used only with the ESCC. Write Register 7 Prime is located at the same address as Write Register 7. This register is written to by setting bit D0 of WR15 to a 1. Refer to the description in the section on Write Register 15. Features enabled in WR7 Prime remain enabled unless otherwise disabled; a hardware or channel reset leaves WR7 Prime with all features intact (register contents are 0) (Figure 5-10).

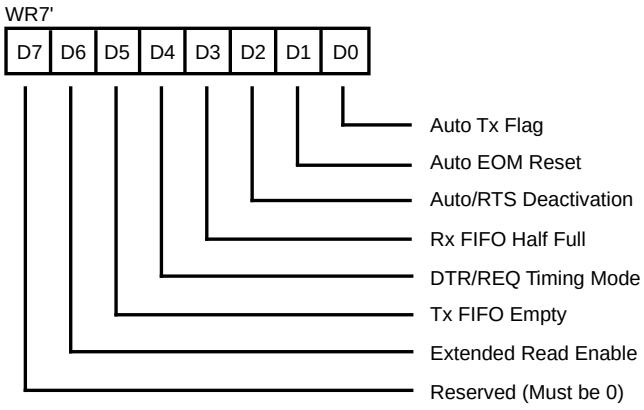



**Figure 5-10. Write Register 7 Prime**

**Bit 7: Reserved**
This bit is not used and must always be written zero.

**Bit 6: Extended Read Enable bit**
Setting this bit enables the reading of WR3, WR4, WR5, WR7 Prime and WR10. When this feature is enabled, these registers can be accessed by reading RR9, RR4, RR5, RR14, and RR11, respectively. When the extended read is not enabled, register access is identical to that of the NMOS/CMOS version. Refer to Chapter Two on how this feature affects the mapping of read registers.

**Bit 5: Transmit FIFO Interrupt Level**
If this bit is set, the transmit buffer empty interrupt is generated when the Transmit FIFO is completely empty. If this bit is reset (0), the transmit buffer empty interrupt is generated when the entry location of the Transmit FIFO is empty. This latter operation is identical to that of the NMOS/CMOS version.

In the DMA Request on Transmit Mode, when using either the /W//REQ or /DTR//REQ pins, the request is asserted when the Transmit FIFO is completely empty if the Transmit FIFO Interrupt Level bit is set. The request is asserted when the entry location of the Transmit FIFO is empty if the Transmit FIFO Interrupt Level bit is reset (0).

**Bit 4: /DTR//REQ Timing**
If this bit is set and the /DTR//REQ pin is used for Request Mode (WR14 bit D2 = 1), the deactivation of the /DTR//REQ pin is identical to the /W//REQ pin. Refer to the chapter on interfacing for further details. If this bit is reset (0), the deactivation time for the /DTR//REQ pin is 4TcPc. This latter operation is identical to that of the SCC.

**Bit 3: Receive FIFO Interrupt Level**
If WR7' D3=1 and "Receive Interrupt on All Characters and Special Conditions" is enabled, the Receive Character Available interrupt is triggered when the Rx FIFO is half full, i.e., the four byte slots of the Rx FIFO are empty. However, if any character has a special condition, a special condition interrupt is generated when the character is loaded into the Receive FIFO. Therefore, the special condition interrupt service routine should read RR1 before reading the data to determine which byte has which special condition.

If WR7' D3=0, the ESCC sets the receiver and generates the receive character available interrupt on every received character, regardless of any special receive condition.

**Bit 2: Auto /RTS pin Deactivation**
This bit controls the timing of the deassertion of the /RTS pin. If the ESCC is programmed for SDLC mode and Flag-On-Underrun (WR10 D2=0), this bit is set and the RTS bit is reset. The /RTS is deasserted automatically at the last bit of the closing flag, triggered by the rising edge of the Transmit Clock. If this bit is reset, the /RTS pin follows the state programmed in WR5 D1.

**Bit 1: Automatic EOM Reset**
If this bit is set, the ESCC automatically resets the Tx Underrun/EOM latch and presets the transmit CRC generator to its programmed preset state (per values set in WR5 D2 & WR10 D7). Therefore, it is not necessary to issue the Reset Tx Underrun/EOM latch command when this feature is enabled. If this bit is reset, ESCC operation is identical to the SCC.

**Bit 0: Automatic Tx SDLC Flag**
If this bit is set, the ESCC automatically transmits an SDLC flag before transmitting data. This removes the requirement to reset the mark idle bit (WR10 D3) before writing data to the transmitter, or having to enable the transmitter before writing data to the Transmit FIFO. Also, this feature enables a transmit data write before enabling the transmitter. If this bit is reset, operation is identical to that of the SCC.

## 5.2.10 Write Register 7 Prime (85C30 only)

This Register is used only with the CMOS 85C30 SCC. WR7' is written to by first setting bit D0 of WR15 to 1, and pointing to WR7 as normal. All writes to register 7 will be to WR7' so long as WR D0 is set. WR 15 bit D0 must be reset to 0 to address the sync register, WR7. If bit D6 of WR7' was set during the write, then WR7' can be read by accessing to RR14. The features remain enabled until specifically disabled, or disabled by a hardware or software reset. Figure 5-10a. shows WR7'.

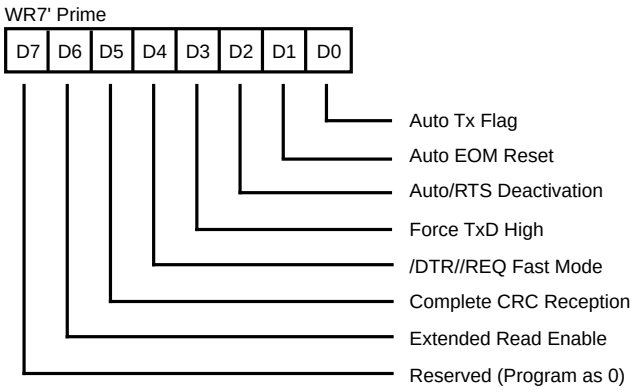



**Figure 5-10a. Write Register 7 Prime (WR7')**

**Bit 7: Reserved.**
This bit is reserved and must be programmed as 0.

**Bit 6: Extended Read Enable bit**
This bit enables the Extended Read. Setting this bit enables the reading of WR3, WR4, WR5, WR7' and WR10. When this feature is enabled, these registers can be accessed by reading RR9, RR4, RR5, RR14, and RR11, respectively. When this feature is not enabled, register access is to the SCC. In this case, read to these register locations returns RR13, RR0, RR1, RR10, and RR15 respectively.

**Bit 5: Receive Complete CRC**
On this version, with this bit set to 1, the 2nd byte of the CRC is received completely. This feature is ideal for applications which require a 2nd CRC byte for complete data; for example, a protocol analyzer or applications using other than CRC-CCITT CRC (i.e., 32bit CRC).

In SDLC mode of operation, the CMOS SCC, on this bit is programmed as 0. In this case on the EOF condition (when the closing flag is detected), the contents of the Receive Shift Register are transferred to the Receive Data FIFO regardless of the number of bits assembled. Because of the three-bit delay path between the sync register and the Receive Shift register, the last two bits of the 2nd byte of the CRC are never transferred to the Receive Data FIFO. The data is actually formed with the six Least Significant Bits of the 2nd CRC byte.

**Bit 4: /DTR//REQ Timing Fast Mode.**
If this bit is set and the /DTR//REQ pin is used for Request Mode (WR14, bit D2=1), the deactivation of the /DTR//REQ pin is identical to the /W//REQ pin, which is triggered on the falling edge of the /WR signal, and the /DTR//REQ pin goes inactive below 200 ns (this number varies depending on the speed grade of the device). When this bit is reset to 0, the deactivation time for the /DTR//REQ pin is 4TcPc.

**Bit 3: Force TxD High.**
In the SDLC mode of operation with the NRZI encoding mode, there is an option to force TxD High. If bit D0 of WR15 is set to 1, bit D3 of WR7' can be used to set TxD pin High.

Note that the operation of this bit is independent of the Tx Enable bit in WR5 is used to control transmission activities, whereas bit D3 of WR7' acts as a pseudo transmitter may actually be mark or flag idling. Care must be exercised when setting this bit because any character being transmitted at the time that bit is set is "chopped off"; data written to the Transmit Buffer while this bit is set is lost.

**Bit 2: Auto /RTS pin Deactivation**
This bit controls the timing of the deassertion of the /RTS pin. If this device is programmed for SDLC mode and Flag-On-Underrun (WR10 D2=0), this bit is set and the RTS bit is reset. The /RTS is deasserted automatically at the last bit of the closing flag, triggered by the rising edge of the TxC. If this bit is reset to 0, the /RTS pin follows the state programmed in WR5 bit D1.

**Bit 1: Automatic Tx Underrun/EOM Latch Reset**
If this bit is set, this version automatically resets the Tx Underrun/EOM latch and presets the transmit CRC generator to its programmed preset state (the values set in WR5 D2 & WR10 D7). This removes the requirement to issue the Reset Tx Underrun/EOM latch command. Also, this feature enables a write transmit data before enabling the transmitter.

**Bit 0: Automatic SDLC Opening Flag Transmission.**
If this bit is set, the device automatically transmits an SDLC opening flag before transmitting data. This removes the requirement to reset the mark idle bit (WR10, bit D3) before writing data to the transmitter, or having to enable the transmitter before writing data to the Transmit buffer. Also, this feature enables a write transmit data before enabling the transmitter.

## 5.2.11 Write Register 8 (Transmit Buffer)

WR8 is the transmit buffer register.

## 5.1 INTRODUCTION (Continued)

### 5.2.12 Write Register 9 (Master Interrupt Control)

WR9 is the Master Interrupt Control register and contains the Reset command bits. Only one WR9 exists in the SCC and is accessed from either channel. The Interrupt control bits are programmed at the same time as the Reset command, because these bits are only reset by a hardware reset. Bit positions for WR9 are shown in Figure 5-11.

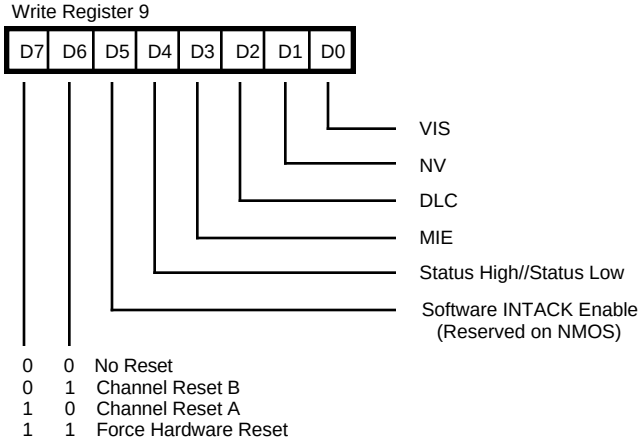



**Figure 5-11. Write Register 9**

**Bit 7 and 6: Reset Command Bits**
Together, these bits select one of the reset commands for the SCC. Setting either of these bits to 1 disables both the receiver and the transmitter in the corresponding channel; forces TxD for that channel marking, forces the modem control signals High in that channel, resets all IPs and IUSs and disables all interrupts in that channel. Four extra PCLK cycles must be allowed beyond the usual cycle time after any of the reset commands is issued before any additional commands or controls are written to the channel affected.

**Null Command (00).** This command has no effect. It is used when a write to WR9 is necessary for some reason other than an SCC Reset command.

**Channel Reset B Command (01).** Issuing this command causes a channel reset to be performed on Channel B.

**Channel Reset A Command (10).** Issuing this command causes a channel reset to be performed on Channel A.

**Force Hardware Reset Command (11).** The effects of this command are identical to those of a hardware reset, except that the Shift Right/Shift Left bit is not changed and the MIE, Status High/Status Low and DLC bits take the programmed values that accompany this command.

**Bit 5: Software Interrupt Acknowledge control bit**
If bit D5 is set, reading Read Register 2 (RR2) results in an interrupt acknowledge cycle to be executed internally. Like a hardware INTACK cycle, a software acknowledge causes the INT pin to return High, the IEO pin to go Low, and sets the IUS latch for the highest priority interrupt pending.

This bit is reserved on NMOS, and always writes as 0.

**Bit 4: Status High//Status Low control bit**
This bit controls which vector bits the SCC modifies to indicate status. When set to 1, the SCC modifies bits V6, V5, and V4 according to Table 5-6. When set to 0, the SCC modifies bits V1, V2, and V3. This bit controls status in both the vector returned during an interrupt acknowledge cycle and the status in RR2B. This bit is reset by a hardware reset.

**Table 5-6. Interrupt Vector Modification**

| V3 | V2 | V1 | Status High/Status Low =0 |
|---|---|---|---|
| V4 | V5 | V6 | Status High/Status Low =1 |
| 0 | 0 | 0 | Ch B Transmit Buffer Empty |
| 0 | 0 | 1 | Ch B External/Status Change |
| 0 | 1 | 0 | Ch B Receive Char. Available |
| 0 | 1 | 1 | Ch B Special Receive Condition |
| 1 | 0 | 0 | Ch A Transmit Buffer Empty |
| 1 | 0 | 1 | Ch A External/Status Change |
| 1 | 1 | 0 | Ch A Receive Char. Available |
| 1 | 1 | 1 | Ch A Special Receive Condition |

**Bit 3: Master Interrupt Enable**
This bit is set to 1 to globally enable interrupts, and cleared to zero to disable interrupts. Clearing this bit to zero forces the IEO pin to follow the state of the IEI pin unless there is an IUS bit set in the SCC. No IUS bit is set after the MIE bit is cleared to zero. This bit is reset by a hardware reset.

**Bit 2: Disable Lower Chain control bit**
The Disable Lower Chain bit is used by the CPU to control the interrupt daisy chain. Setting this bit to 1 forces the IEO pin Low, preventing lower priority devices on the daisy chain from requesting interrupts. This bit is reset by a hardware reset.

**Bit 1: No Vector select bit**
The No Vector bit controls whether or not the SCC responds to an interrupt acknowledge cycle. This is done by placing a vector on the data bus if the SCC is the highest priority device requesting an interrupt. If this bit is set, no vector is returned; i.e., AD7-AD0 remains tri-stated during an interrupt acknowledge cycle, even if the SCC is the highest priority device requesting an interrupt.

**Bit 0: Vector Includes Status control bit**
The Vector Includes Status Bit controls whether or not the SCC includes status information in the vector it places on the bus in response to an interrupt acknowledge cycle. If this bit is set, the vector returned is variable, with the variable field depending on the highest priority IP that is set. Table 5-5 shows the encoding of the status information. This bit is ignored if the No Vector (NV) bit is set.

## 5.2.13 Write Register 10 (Miscellaneous Transmitter/Receiver Control Bits)

WR10 contains miscellaneous control bits for both the receiver and the transmitter. Bit positions for WR10 are shown in Figure 5-12. On the ESCC and 85C30 with the Extended Read option enabled, this register may be read as RR11.

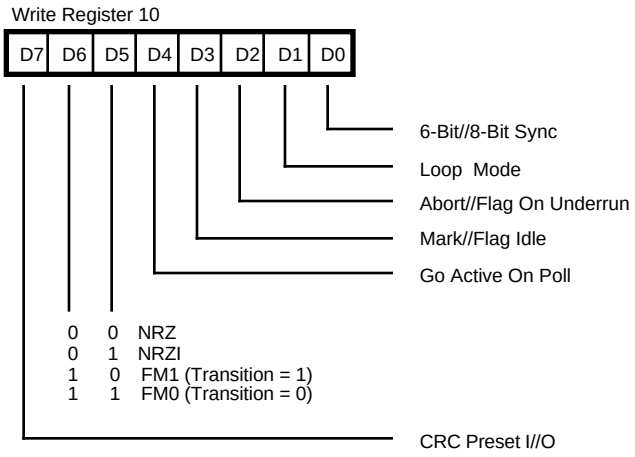



**Figure 5-12. Write Register 10**

**Bit 7: CRC Presets I/O select bit**
This bit specifies the initialized condition of the receive CRC checker and the transmit CRC generator. If this bit is set to 1, the CRC generator and checker are preset to 1. If this bit is set to 0, the CRC generator and checker are preset to 0. Either option can be selected with either CRC polynomial. In SDLC mode, the transmitted CRC is inverted before transmission, and the received CRC is checked against the bit pattern 0001110100001111. This bit is reset by a channel or hardware reset. This bit is ignored in Asynchronous mode.

**Bits 6 and 5: Data Encoding select bits.**
These bits control the coding method used for both the transmitter and the receiver, as illustrated in Table 5-7. All of the clocking options are available for all coding methods. The DPLL in the SCC is useful for recovering clocking information in NRZI and FM modes. Any coding method can be used in X1 mode. A hardware reset forces NRZ mode. Timing for the various modes is shown in Figure 5-13.

**Table 5-7. Data Encoding**

| Bit 6 | Bit 5 | Encoding |
|---|---|---|
| 0 | 0 | NRZ |
| 0 | 1 | NRZI |
| 1 | 0 | FM1 (transition = 1) |
| 1 | 1 | FM0 (transition = 0) |

## 5.1 INTRODUCTION (Continued)

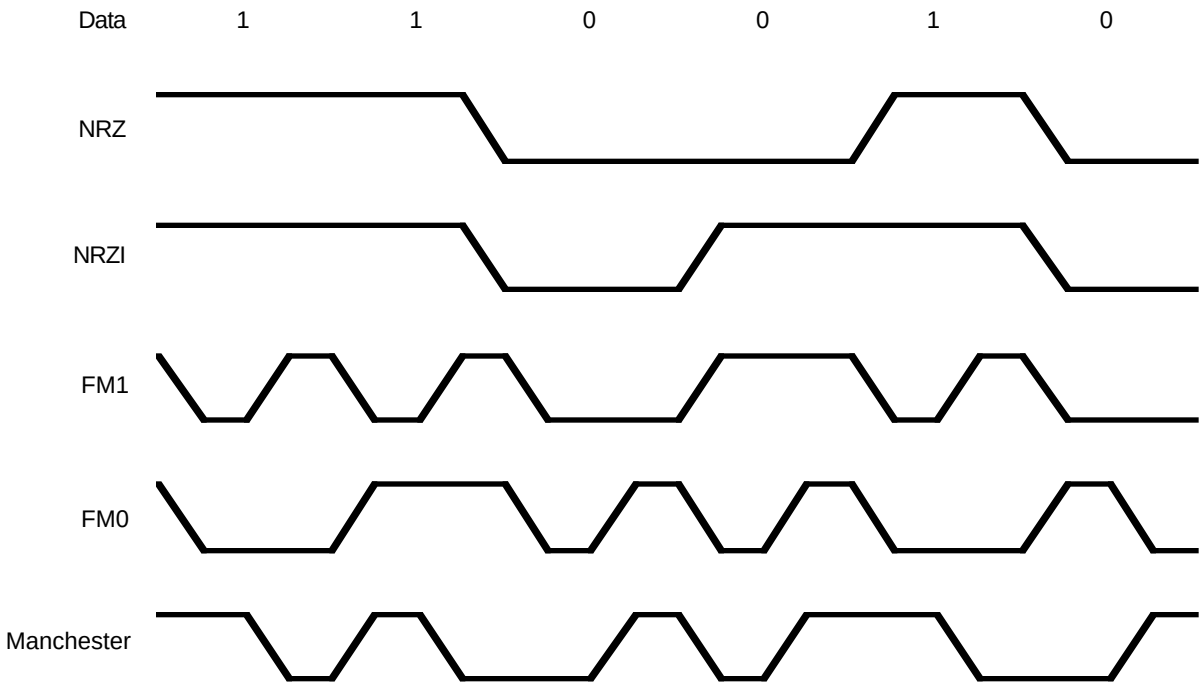



**Figure 5-13. NRZ (NRZI), FM1 (FM0) Timing**

**Bit 4: Go-Active-On-Poll control bit**
When Loop mode is first selected during SDLC operation, the SCC connects RxD to TxD with only gate delays in the path. The SCC does not go on-loop and insert the 1-bit delay between RxD and TxD until this bit has been set and an EOP received. When the SCC is on-loop, the transmitter does not go active unless this bit is set at the time an EOP is received. The SCC examines this bit whenever the transmitter is active in SDLC Loop mode and is sending a flag. If this bit is set at the time the flag is leaving the Transmit Shift register, another flag or data byte (if the transmit buffer is full) is transmitted.

If the Go-Active-On-Poll bit is not set at this time, the transmitter finishes sending the flag and reverts to the 1-Bit Delay mode. Thus, to transmit only one response frame, this bit is reset after the first data byte is sent to the SCC, but before CRC has been transmitted. If the bit is not reset before CRC is transmitted, extra flags are sent, slowing down response time on the loop. If this bit is reset before the first data is written, the SCC completes the transmission of the present flag and reverts to the 1-Bit Delay mode.

After gaining control of the loop, the SCC is not able to transmit again until a flag and another EOP are received. It is good practice to set this bit only upon receipt of a poll frame to ensure that the SCC does not go on-loop without the CPU noticing it.

In synchronous modes other than SDLC with the Loop Mode bit set, this bit is set before the transmitter goes active in response to a received sync character.

This bit is always ignored in Asynchronous mode and Synchronous modes unless the Loop Mode bit is set. This bit is reset by a channel or hardware reset.

**Bit 3: Mark//Flag Idle line control bit**
This bit affects only SDLC operation and is used to control the idle line condition. If this bit is set to 0, the transmitter send flags as an idle line. If this bit is set to 1, the transmitter sends continuous 1s after the closing flag of a frame. The idle line condition is selected byte by byte i.e., either a flag or eight 1s are transmitted. The primary station in an SDLC loop should be programmed for Mark Idle to create the EOP sequence. Mark Idle must be deselected at the beginning of a frame before the first data is written to the SCC, so that an opening flag is transmitted. This bit is ignored in Loop mode, but the programmed value takes effect upon exiting the Loop mode. This bit is reset by a channel or hardware reset.

On the ESCC and 85C30 with the Automatic TX SDLC Flag mode enabled (WR7', D0=1), this bit can be left as mark idle. It will send an opening flag automatically, as well as sending a closing flag followed by mark idle after the frame transmission is completed.

**Bit 2: Abort//Flag On Underrun select bit**
This bit affects only SDLC operation and is used to control how the SCC responds to a transmit underrun condition. If this bit is set to 1 and a transmit underrun occurs, the SCC sends an abort and a flag instead of a CRC. If this bit is reset, the SCC sends a CRC on a transmit underrun. At the beginning of this 16-bit transmission, the Transmit Underrun/EOM bit is set, causing an External/Status interrupt. The CPU uses this status, along with the byte count from memory or the DMA, to determine whether the frame must be retransmitted.

To start the next frame, a Transmit Buffer Empty interrupt occurs at the end of this 16-bit transmission. If both this bit and the Mark/Flag Idle bit are set to 1, all 1s are transmitted after the transmit underrun. This bit should be set after the first byte of data is sent to the SCC and reset immediately after the last byte of data, terminating the frame properly with CRC and a flag. This bit is ignored in Loop mode, but the programmed value is active upon exiting Loop mode. This bit is reset by a channel or hardware reset.

**Bit 1: Loop Mode control bit**
In SDLC mode, the initial set condition of this bit forces the SCC to connect TxD to RxD and to begin searching the incoming data stream so that it can go on loop. All bits pertinent to SDLC mode operation in other registers are set before this mode is selected. The transmitter and receiver are not enabled until after this mode has been selected. As soon as the Go-Active-On-Poll bit is set and an EOP is received, the SCC goes on-loop. If this bit is reset after the SCC goes on-loop, the SCC waits for the next EOP to go off-loop.

In synchronous modes, the SCC uses this bit, along with the Go-Active-On-Poll bit, to synchronize the transmitter to the receiver. The receiver should not be enabled until after this mode is selected. The TxD pin is held marking when this mode is selected unless a break condition is programmed. The receiver waits for a sync character to be received and then enables the transmitter on a character boundary. The break condition, if programmed, is removed. This mode works properly with sync characters of 6, 8, or 16 bits. This bit is ignored in Asynchronous mode and is reset by a channel or hardware reset.

**Bit 0: 6-Bit/8-Bit SYNC select bit**
This bit is used to select a special case of synchronous modes. If this bit is set to 1 in Monosync mode, the receiver and transmitter sync characters are six bits long instead of the usual eight. If this bit is set to 1 in Bisync mode, the received sync is 12 bits and the transmitter sync character remains 16 bits long. This bit is ignored in SDLC and Asynchronous modes, but still has effect in the special external sync modes. This bit is reset by a channel or hardware reset.

## 5.2.14 Write Register 11 (Clock Mode Control)

WR11 is the Clock Mode Control register. The bits in this register control the sources of both the receive and transmit clocks, the type of signal on the /SYNC and /RTxC pins, and the direction of the /TRxC pin. Bit positions for WR11 are shown in Figure 5-14; also, refer to Section 3.5 Clock Selection.

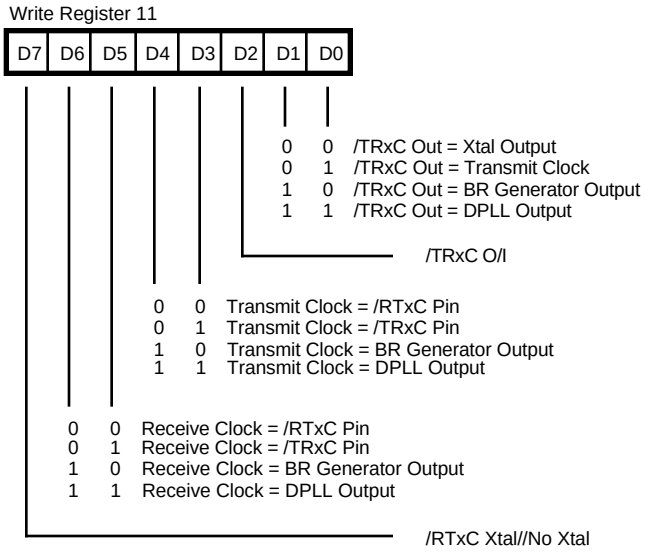



**Figure 5-14. Write Register 11**

**Bit 7: RTxC-XTAL//NO XTAL select bit**
This bit controls the type of input signal the SCC expects to see on the /RTxC pin. If this bit is set to 0, the SCC expects a TTL-compatible signal as an input to this pin. If this bit is set to 1, the SCC connects a high-gain amplifier between the /RTxC and /SYNC pins in expectation of a quartz crystal being placed across the pins.

The output of this oscillator is available for use as a clocking source. In this mode of operation, the /SYNC pin is unavailable for other use. The /SYNC signal is forced to zero internally. A hardware reset forces /NO XTAL. (At least 20 ms should be allowed after this bit is set to allow the oscillator to stabilize.)

**Bits 6 and 5: Receiver Clock select bits 1 and 0**
These bits determine the source of the receive clock as shown in Table 5-8. They do not interfere with any of the modes of operation in the SCC, but simply control a multiplexer just before the internal receive clock input. A hardware reset forces the receive clock to come from the /RTxC pin.

## 5.1 INTRODUCTION (Continued)

**Table 5-8. Receive Clock Source**

| Bit 6 | Bit 5 | Receive Clock |
|---|---|---|
| 0 | 0 | /RTxC Pin |
| 0 | 1 | /TRxC Pin |
| 1 | 0 | BR Output |
| 1 | 1 | DPLL Output |

**Bits 4 and 3: Transmit Clock select bits 1 and 0.**
These bits determine the source of the transmit clock as shown in Table 5-9. They do not interfere with any of the modes of operation of the SCC, but simply control a multiplexer just before the internal transmit clock input. The DPLL output that is used to feed the transmitter in FM modes lags by 90 degrees the output of the DPLL used by the receiver. This makes the received and transmitted bit cells occur simultaneously, neglecting delays. A hardware reset selects the /TRxC pin as the source of the transmit clocks.

**Table 5-9. Transmit Clock Source**

| Bit 4 | Bit 3 | Transmit Clock |
|---|---|---|
| 0 | 0 | /RTxC Pin |
| 0 | 1 | /TRxC Pin |
| 1 | 0 | BR Output |
| 1 | 1 | DPLL Output |

**Bit 2: TRxC Pin I/O control bit**
This bit determines the direction of the /TRxC pin. If this bit is set to 1, the /TRxC pin is an output and carries the signal selected by D1 and D0 of this register. However, if either the receive or the transmit clock is programmed to come from the /TRxC pin, /TRxC is an input, regardless of the state of this bit. The /TRxC pin is also an input if this bit is set to 0. A hardware reset forces this bit to 0.

**Bits 1 and 0: /TRxC Output Source select bits 1 and 0**
These bits determine the signal to be echoed out of the SCC via the /TRxC pin as given in Table 5-10. No signal is produced if /TRxC has been programmed as the source of either the receive or the transmit clock. If /TRxC O/I (bit 2) is set to 0, these bits are ignored.

If the XTAL oscillator output is programmed to be echoed, and the XTAL oscillator is not enabled, the /TRxC pin goes High. The DPLL signal that is echoed is the DPLL signal used by the receiver. Hardware reset selects the XTAL oscillator as the output source.

**Table 5-10. Transmit External Control Selection**

| Bit 1 | Bit 0 | TRxC Pin Output |
|---|---|---|
| 0 | 0 | XTAL Oscillator Output |
| 0 | 1 | Transmit Clock |
| 1 | 0 | BR Output |
| 1 | 1 | DPLL Output (receive) |

### 5.2.15 Write Register 12 (Lower Byte of Baud Rate Generator Time Constant)

WR12 contains the lower byte of the time constant for the baud rate generator. The time constant can be changed at any time, but the new value does not take effect until the next time the time constant is loaded into the down counter. No attempt is made to synchronize the loading of the time constant into WR12 and WR13 with the clock driving the down counter. For this reason, it is advisable to disable the baud rate generator while the new time constant is loaded into WR12 and WR13. Ordinarily, this is done anyway to prevent a load of the down counter between the writing of the upper and lower bytes of the time constant.

The formula for determining the appropriate time constant for a given baud is shown below, with the desired rate in bits per second and the BR clock period in seconds. This formula is derived because the counter decrements from N down to zero-plus-one-cycle for reloading the time constant. This is then fed to a toggle flip-flop to make the output a square wave. Bit positions for WR12 are shown in Figure 5-15.

$$\text{Time Constant} = \frac{\text{Clock Frequency}}{2 \times (\text{Desired Rate}) \times (\text{BR Clock Period})} - 2$$

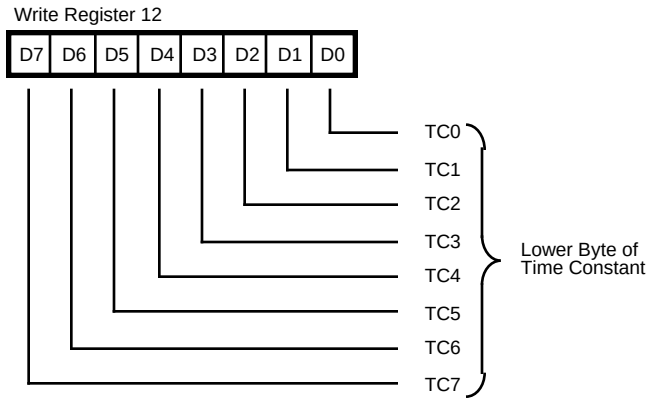



**Figure 5-15. Write Register 12**

## 5.2.16 Write Register 13 (Upper Byte of Baud Rate Generator Time Constant)

WR13 contains the upper byte of the time constant for the baud rate generator. Bit positions for WR13 are shown in Figure 5-16.

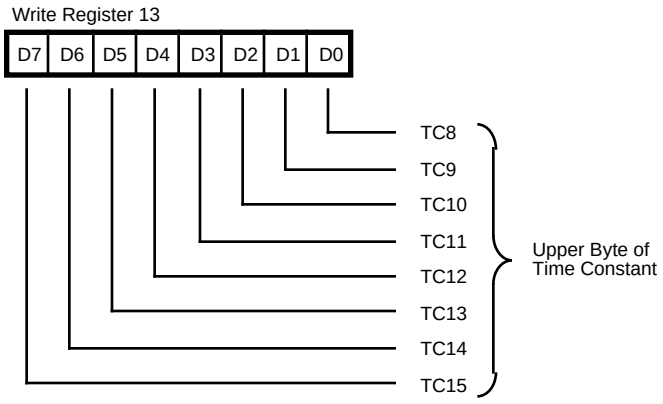



**Figure 5-16. Write Register 13**

## 5.2.17 Write Register 14 (Miscellaneous Control Bits)

WR14 contains some miscellaneous control bits. Bit positions for WR14 are shown in Figure 5-17. For DPLL function, refer to section 3.4 as well.

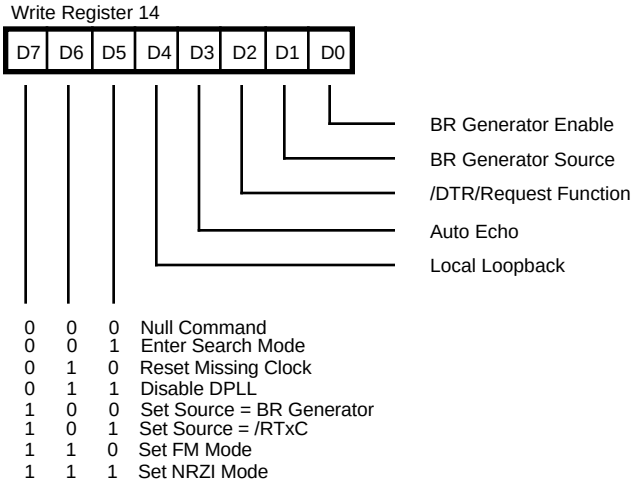



**Figure 5-17. Write Register 14**

**Bits D7-D5: Digital Phase-Locked Loop Command Bits.**

These three bits encode the eight commands for the Digital Phase-Locked Loop. A channel or hardware reset disables the DPLL, resets the missing clock latches, sets the source to the /RTxC pin and selects NRZI mode. The Enter Search Mode command enables the DPLL after a reset.

**Null Command (000).** This command has no effect on the DPLL.

**Enter Search Mode Command (001)**. Issuing this command causes the DPLL to enter the Search mode, where the DPLL searches for a locking edge in the incoming data stream. The action taken by the DPLL upon receipt of this command depends on the operating mode of the DPLL.

In NRZI mode, the output of the DPLL is High while the DPLL is waiting for an edge in the incoming data stream. After the Search mode is entered, the first edge the DPLL sees is assumed to be a valid data edge, and the DPLL begins the clock recovery operation from that point. The DPLL clock rate must be 32x the data rate in NRZI mode. Upon leaving the Search mode, the first sampling edge of the DPLL occurs 16 of these 32x clocks after the first data edge, and the second sampling occurs 48 of these 32x clocks after the first data edge. Beyond this point, the DPLL begins normal operation, adjusting the output to remain in sync with the incoming data.

In FM mode, the output of the DPLL is Low while the DPLL is waiting for an edge in the incoming data stream. The first edge the DPLL detects is assumed to be a valid clock edge. For this to be the case, the line must contain only clock edges; i.e. with FM1 encoding, the line must be continuous 0s. With FM0 encoding the line must be continuous 1s, whereas Manchester encoding requires alternating 1s and 0s on the line. The DPLL clock rate must be 16 times the data rate in FM mode. The DPLL output causes the receiver to sample the data stream in the nominal center of the two halves of the bit to decide whether the data was a 1 or a 0.

After this command is issued, as in NRZI mode, the DPLL starts sampling immediately after the first edge is detected. (In FM mode, the DPLL examines the clock edge of every other bit to decide what correction must be made to remain in sync.) If the DPLL does not see an edge during the expected window, the one clock missing bit in RR10 is set. If the DPLL does not see an edge after two successive attempts, the two clocks missing bits in RR10 are set and the DPLL automatically enters the Search mode. This command resets both clocks missing latches.

**Reset Clock Missing Command (010).** Issuing this command disables the DPLL, resets the clock missing latches in RR10, and forces a continuous Search mode state.

**Disable DPLL Command (011).** Issuing this command disables the DPLL, resets the clock missing latches in RR10, and forces a continuous Search mode state.

## 5.1 INTRODUCTION (Continued)

**Set Source to BRG Command (100).** Issuing this command forces the clock for the DPLL to come from the output of the BRG.

**Set Source to /RTxC Command (101).** Issuing the command forces the clock for the DPLL to come from the /RTxC pin or the crystal oscillator, depending on the state of the XTAL/no XTAL bit in WR11. This mode is selected by a channel or hardware reset.

**Set FM Mode Command (110).** This command forces the DPLL to operate in the FM mode and is used to recover the clock from FM or Manchester-Encoded data. (Manchester is decoded by placing the receiver in NRZ mode while the DPLL is in FM mode.)

**Set NRZI Mode Command (111).** Issuing this command forces the DPLL to operate in the NRZI mode. This mode is also selected by a hardware or channel reset.

**Bit 4: Local Loopback select bit**
Setting this bit to 1 selects the Local Loopback mode of operation. In this mode, the internal transmitted data is routed back to the receiver, and to the TxD pin. The /CTS and /DCD inputs are ignored as enables in Local Loopback mode, even if auto enable is selected. (If so programmed, transitions on these inputs still cause interrupts.) This mode works with any Transmit/Receive mode except Loop mode. For meaningful results, the frequency of the transmit and receive clocks must be the same. This bit is reset by a channel or hardware reset.

**Bit 3: Auto Echo select bit**
Setting this bit to 1 selects the Auto Echo mode of operation. In this mode, the TxD pin is connected to RxD as in Local Loopback mode, but the receiver still listens to the RxD input. Transmitted data is never seen inside or outside the SCC in this mode, and /CTS is ignored as a transmit enable. This bit is reset by a channel or hardware reset.

**Bit 2: DTR/Request Function select bit**
This bit selects the function of the /DTR//REQ pin following the state of the DTR bit in WR5. If this is set to 0, the /DTR//REQ pin follows the state of the DTR bit in WR5. If this bit is set to 1, the /DTR//REQ pin goes Low whenever the transmit buffer becomes empty and in any of the synchronous modes when the CRC has been sent at the end of a message. The request function on the /DTR//REQ pin differs from the transmit request function available on the /W//REQ pin. The /REQ does not go inactive until the internal operation satisfying the request is complete, which occurs three to four PCLK cycles after the falling edge of /DS, /RD or /WR. If the DMA used is edge-triggered, this difference is unimportant. The deassertion timing of the REQ mode can be programmed to occur with the same timing as the /W/REQ pin if WR7' D4=1. This bit is reset by a channel or hardware reset.

**Bit 1: Baud Rate Generator Source select bit**
This bit selects the source of the clock for the baud rate generator, If this bit is set to 0. The baud rate generator clock comes from either the /RTxC pin or the XTAL oscillator (depending on the state of the XTAL//no XTAL bit). If this bit is set to 1, the clock for the baud rate generator is the SCC's PCLK input. Hardware reset sets this bit to 0, select the /RTxC pin as the clock source for the BRG.

**Bit 0: Baud Rate Generator Enable**
This bit controls the operation of the BRG. The counter in the BRG is enabled for counting when this bit is set to 1, and counting is inhibited when this bit is set to 0. When this bit is set to 1, change in the state of this bit is not reflected by the output of the BRG for two counts of the counter. This allows the command to be synchronized. However, when set to 0, disabling is immediate. This bit is reset by a hardware reset.

### 5.2.18 Write Register 15 (External/Status Interrupt Control)

WR15 is the External/Status Source Control register. If the External/Status interrupts are enabled as a group via WR1, bits in this register control which External/Status conditions cause an interrupt. Only the External/Status conditions that occur after the controlling bit is set to 1 cause an interrupt. This is true, even if an External/Status condition is pending at the time the bit is set. Bit positions for WR15 are shown in Figure 5-18.

On the CMOS version, bits D2 and D0 are reserved. On the NMOS version, bit D2 is reserved. These reserved bits should be written as 0s.

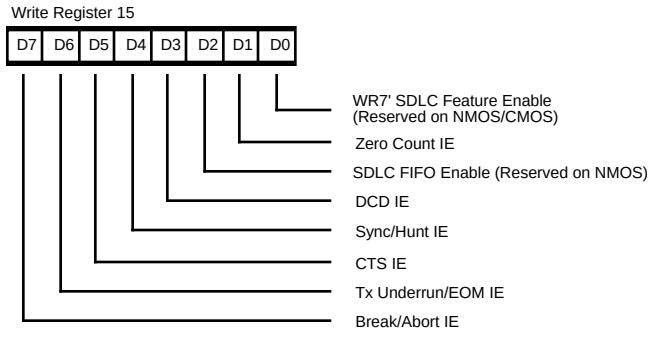



**Figure 5-18. Write Register 15**

**Bit 7: Brea/Abort Interrupt Enable**
If this bit is set to 1, a change in the Break/Abort status of the receiver causes an External/Status interrupt. This bit is set by a channel or hardware reset.

**Bit 6: Transmit Underrun/EOM Interrupt Enable**
If this bit is set to 1, a change of state by the Tx Underrun/EOM latch in the transmitter causes an External/Status interrupt. This bit is set to 1 by a channel or hardware reset.

**Bit 5: CTS Interrupt Enable**
If this bit is set to 1, a change of state on the /CTS pin causes an External/Status Interrupt. This bit is set by a channel or hardware reset.

**Bit 4: SYNC/Hunt Interrupt Enable**
If this bit is set to 1, a change of state on the /SYNC pin causes an External/Status interrupt in Asynchronous mode, and a change of state in the Hunt bit in the receiver causes and External/Status interrupt in synchronous modes. This bit is set by a channel or hardware reset.

**Bit 3: DCD Interrupt Enable**
If this bit is set to 1, a change of state on the /DCD pin causes an External/Status interrupt. This bit is set by a channel or hardware reset.

**Bit 2: Status FIFO Enable control bit (CMOS/ESCC)**
If this bit is set and if the CMOS/ESCC is in the SDLC/HDLC Mode, status (five bits from Read Register 1: Residue, Overrun, and CRC Error) and fourteen bits of byte count are held in the Status FIFO until read. Status information for up to ten frames can be stored. If this bit is reset (0) or if the CMOS/ESCC is not in the SDLC/HDLC Mode, the FIFO is not operational and status information read reflects the current status only. This bit is reset to 0 by a channel or hardware reset. For details on this function, refer to Section 4.4.3.

On the NMOS version, this bit is reserved and should be programmed as 0.

**Bit 1: Zero Count Interrupt Enable**
If this bit is set to 1, an External/Status interrupt is generated whenever the counter in the baud rate generator reaches 0. This bit is reset to 0 by a channel or hardware reset.

**Bit 0: Point to Write Register WR7 Prime (ESCC and 85C30 only)**
When this bit is programmed to 0, writes to the WR7 address are made to WR7. When this bit is programmed to 1, writes to the WR7 address are made to WR7 Prime. Once set, this bit remains set unless cleared by writing a 0 to this bit or by a hardware or software reset. Note that if the extended read option is enabled, WR7 Prime is read in RR14. For details about WR7', refer to Section 4.4.1.2 and Section 5.2.9.

On the NMOS/CMOS version, this bit is reserved and should be programmed as 0.

## 5.3 READ REGISTERS

The SCC Read register set in each channel has four status registers (includes receive data FIFO), and two baud rate time constant registers in each channel. The Interrupt Vector register (RR2) and Interrupt Pending register (RR3) are shared by both channels. In addition to these, the CMOS/ESCC has two additional registers for the SDLC Frame Status FIFO. On the ESCC, if that function is enabled (WR7' bit D6=1), five more registers are available which return the value written to the write registers.

The status of these registers is continually changing and depends on the mode of communication, received and transmitted data, and the manner in which this data is transferred to and from the CPU. The following description details the bit assignment for each register.

### 5.3.1 Read Register 0 (Transmit/Receive Buffer Status and External Status)

Read Register 0 (RR0) contains the status of the receive and transmit buffers. RR0 also contains the status bits for the six sources of External/Status interrupts. The bit configuration is illustrated in Figure 5-19.

On the NMOS/CMOS version, note that the status of this register might be changing during the read.

An enhancement allows the ESCC and 85C30 to latch the contents of RR0 during read transactions for this register. The latch is released on the rising edge of the /RD of the read transaction to this register. This feature prevents missed status due to changes that take place when the read cycle is in progress.

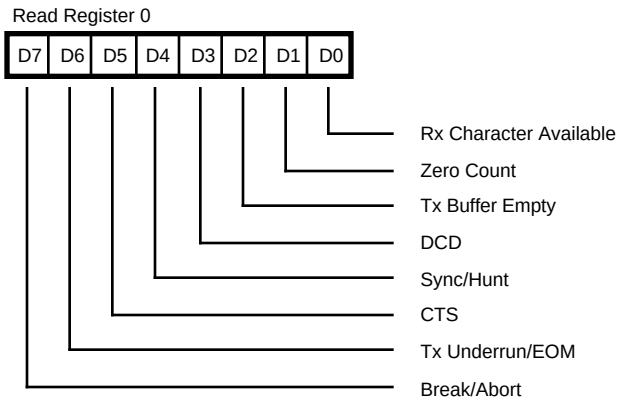



**Figure 5-19. Read Register 0**

## 5.3 READ REGISTERS (Continued)

**Bit 7: Break/Abort status**
In the Asynchronous mode, this bit is set when a Break sequence (null character plus framing error) is detected in the receive data stream. This bit is reset when the sequence is terminated, leaving a single null character in the Receive FIFO. This character is read and discarded. In SDLC mode, this bit is set by the detection of an Abort sequence (seven or more 1s), then reset automatically at the termination of the Abort sequence. In either case, if the Break/Abort IE bit is set, an External/Status interrupt is initiated. Unlike the remainder of the External/Status bits, both transitions are guaranteed to cause an External/Status interrupt, even if another External/Status interrupt is pending at the time these transitions occur. This procedure is necessary because Abort or Break conditions may not persist.

**Bit 6: Transmit Underrun/EOM status**
This bit is set by a channel or hardware reset when the transmitter is disabled or a Send Abort command is issued. This bit is only reset by the reset Tx Underrun/EOM Latch command in WR0. When the Transmit Underrun occurs, this bit is set and causes an External/Status interrupt (if the Tx Underrun/EOM IE bit is set).

Only the 0-to-1 transition of this bit causes an interrupt. This bit is always 1 in Asynchronous mode, unless a reset Tx Underrun/EOM Latch command has been erroneously issued. In this case, the Send Abort command can be used to set the bit to one and at the same time cause an External/Status interrupt.

**Bit 5: Clear to Send pin status**
If the CTS IE bit in WR15 is set, this bit indicates the state of the /CTS pin while no interrupt is pending, latches the state of the /CTS pin and generates an External/Status interrupt. Any odd number of transitions on the /CTS pin causes another External/Status interrupt condition. If the CTS IE bit is reset, it merely reports the current unlatched state of the /CTS pin.

**Bit 4: Sync/Hunt status**
The operation of this bit is similar to that of the CTS bit, except that the condition monitored by the bit varies depending on the mode in which the SCC is operating.

When the XTAL oscillator option is selected in asynchronous modes, this bit is forced to 0 (no External/Status interrupt is generated). Selecting the XTAL oscillator in synchronous or SDLC modes has no effect on the operation of this bit.

The XTAL oscillator should not be selected in External Sync mode.

In Asynchronous mode, the operation of this bit is identical to that of the CTS status bit, except that this bit reports the state of the /SYNC pin.

In External sync mode the /SYNC pin is used by external logic to signal character synchronization. When the Enter Hunt Mode command is issued in External Sync mode, the /SYNC pin must be held High by the external sync logic until character synchronization is achieved. A High on the /SYNC pin holds the Sync/Hunt bit in the reset condition.

When external synchronization is achieved, /SYNC is driven Low on the second rising edge of the Receive Clock after the last rising edge of the Receive Clock on which the last bit of the receive character was received. Once /SYNC is forced Low, it is good practice to keep it Low until the CPU informs the external sync logic that synchronization is lost or that a new message is about to start. Both transitions on the /SYNC pin cause External/Status interrupts if the Sync/Hunt IE bit is set to 1.

The Enter Hunt Mode command should be issued whenever character synchronization is lost. At the same time, the CPU should inform the external logic that character synchronization has been lost and that the SCC is waiting for /SYNC to become active.

In the Monosync and Bisync Receive modes, the Sync/Hunt status bit is initially set to 1 by the Enter Hunt Mode command. The Sync/Hunt bit is reset when the SCC established character synchronization. Both transitions cause External/Status interrupts if the Sync/Hunt IE bit is set. When the CPU detects the end of message or the loss of character synchronization, the Enter Hunt Mode command should be issued to set the Sync/Hunt bit and cause an External/Status interrupt. In this mode, the /SYNC pin is an output, which goes Low every time a sync pattern is detected in the data stream.

In the SDLC modes, the Sync/Hunt bit is initially set by the Enter Hunt Mode command or when the receiver is disabled. It is reset when the opening flag of the first frame is detected by the SCC. An External/Status interrupt is also generated if the Sync/Hunt IE bit is set. Unlike the Monosync and Bisync modes, once the Sync/Hunt bit is reset in SDLC mode, it does not need to be set when the end of the frame is detected. The SCC automatically maintains synchronization. The only way the Sync/Hunt bit is set again is by the Enter Hunt Mode command or by disabling the receiver.

**Bit 3: Data Carrier Detect status**
If the DCD IE bit in WR15 is set, this bit indicates the state of the /DCD pin the last time the Enabled External/Status bits changed. Any transition on the /DCD pin, while no interrupt is pending, latches the state of the /DCD pin and

generates an External/Status interrupt. Any odd number of transitions on the /DCD pin while another External/Status interrupt condition. If the DCD IE is reset, this bit merely reports the current, unlatched state of the /DCD pin.

**Bit 2: TX Buffer Empty status**
This bit is set to 1 when the transmit buffer is empty. It is reset while the CRC is sent in a synchronous or SDLC mode and while the transmit buffer is full. The bit is reset when a character is loaded into the transmit buffer.

On the ESCC, the status of this bit is not related to the Transmit Interrupt Status or the state of WR7' bit D5, but it shows the status of the entry location of the Transmit FIFO. This means more data can be written without being overwritten. This bit is set to 1 when the entry location of the Transmit FIFO is empty. It is reset when a character is loaded into the entry location of the Transmit FIFO.

This bit is always in the set condition after a hardware or channel reset.

For more information on this bit, refer to Section 2.4.8 "Transmit Interrupts and Transmit Buffer Empty bit".

**Bit 1: Zero Count status**
If the Zero Count interrupt Enable bit is set in WR15, this bit is set to one while the counter in the baud rate generator is at the count of zero. If there is no other External/Status interrupt condition pending at the time this bit is set, an External/Status interrupt is generated. However, if there is another External/Status interrupt pending at this time, no interrupt is initiated until interrupt service is complete. If the Zero Count condition does not persist beyond the end of the interrupt service routine, no interrupt is generated. This bit is not latched High, even though the other External/Status latches close as a result of the Low-to-High transition on ZC. The interrupt routine checks the other External/Status conditions for changes. If none changed, ZC was the source. In polled applications, check the IP bit in RR3A for a status change and then proceed as in the interrupt service routine.

**Bit 0: Receive Character Available**
This bit is set to 1 when at least one character is available in the receive data FIFO. It is reset when the receive data FIFO is completely empty. A channel or hardware reset empties the receive data FIFO.

On the ESCC, the status of this bit is independent of WR7' bit D3.

For details on this bit, refer to Section 2.4.7, The Receive Interrupt.

## 5.3.2 Read Register 1

RR1 contains the Special Receive Condition status bits and the residue codes for the I-field in SDLC mode. Figure 5-20 shows the bit positions for RR1.

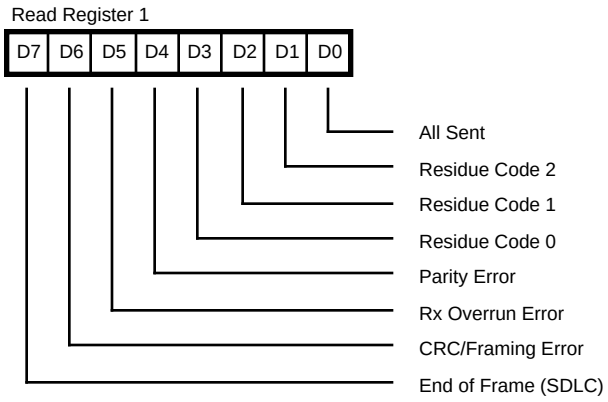



**Figure 5-20. Read Register 1**

**Bit 7: End of Frame (SDLC) status**
This bit is used only in SDLC mode and indicates that a valid closing flag has been received and that the CRC Error bit and residue codes are valid. This bit is reset by issuing the Error Reset command. It is also updated by the first character of the following frame. This bit is reset in any mode other than SDLC.

**Bit 6: CRC/Framing Error status**
If a framing error occurs (in Asynchronous mode), this bit is set (and not latched) for the receive character in which the framing error occurred. Detection of a framing error adds an additional one-half bit to the character time so that the framing error is not interpreted as a new Start bit. In Synchronous and SDLC modes, this bit indicates the result of comparing the CRC checker to the appropriate check value. This bit is reset by issuing an Error Reset command, but the bit is never latched. Therefore, it is always updated when the next character is received. When used for CRC error status in Synchronous or SDLC modes, this bit is usually set since most bit combinations, except for a correctly completed message, result in a non-zero CRC.

On the CMOS and ESCC, if the Status FIFO is enabled (refer to the description in Write Register 15, bit D2 and the description in Read Register 7, bits D7 and D6), this bit reflects the status stored at the exit location of the Status FIFO.

**Bit 5: Receiver Overrun Error status**
This bit indicates that the Receive FIFO has overflowed. Only the character that has been written over is flagged with this error. When that character is read, the Error condition is latched until reset by the Error Reset command.

## 5.3 READ REGISTERS (Continued)

Also, a Special Receive Condition vector is returned, caused by the overrun characters and all subsequent characters received until the Error Reset command is issued.

On the CMOS and ESCC, if the Status FIFO is enabled (refer to the description in Write Register 15, bit D2 and the description in Read Register 7, bits D7 and D6), this bit reflects the status stored at the exit location of the Status FIFO.

**Bit 4: Parity Error status.**
When parity is enabled, this bit is set for the characters whose parity does not match the programmed sense (even/odd). This bit is latched so that once an error occurs, it remains set until the Error Reset command is issued. If the parity in Special Condition bit is set, a parity error causes a Special Receive Condition vector to be returned on the character containing the error and on all subsequent characters until the Error Reset command is issued.

**Bits 3, 2, and 1: Residue Codes, bits 2, 1, and 0**
In those cases in SDLC mode where the received I-Field is not an integral multiple of the character length, these three bits indicate the length of the I-Field and are meaningful only for the transfer in which the end of frame bit is set. This field is set to 011 by a channel or hardware reset and is forced to this state in Asynchronous mode. These three bits can leave this state only if SDLC is selected and a character is received. The codes signify the following (Reference Table 5-11) when a receive character length is eight bits per character.

On the CMOS and ESCC, if the Status FIFO is enabled (refer to the description in Write Register 15, bit D2 and the description in Read Register 7, bits D7 and D6), these bits reflect the status stored at the exit location of the Status FIFO.

I-Field bits are right-justified in all cases. If a receive character length other than eight bits is used for the I-Field, a table similar to Table 5-11 can be constructed for each different character length. Table 5-12 shows the residue codes for no residue (The I-Field boundary lies on a character boundary).

**Table 5-11. I-Field Bit Selection (8 Bits Only)**

| Bit 3 | Bit 2 | Bit 1 | I-Field Bits in Last Byte | I-Field Bits in Previous Byte |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | 3 |
| 0 | 1 | 0 | 0 | 4 |
| 1 | 1 | 0 | 0 | 5 |
| 0 | 0 | 1 | 0 | 6 |
| 1 | 0 | 1 | 0 | 7 |
| 0 | 1 | 1 | 0 | 8 |
| 1 | 1 | 1 | 1 | 8 |
| 0 | 0 | 0 | 2 | 8 |

**Table 5-12. Bits per Character Residue Decoding**

| Bits per Character | Bit 3 | Bit 2 | Bit 1 |
|---|---|---|---|
| 8 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 |
| 6 | 0 | 1 | 0 |
| 5 | 0 | 0 | 1 |

**Bit 0: All Sent status**
In Asynchronous mode, this bit is set when all characters have completely cleared the transmitter pins. Most modems contain additional delays in the data path, which requires the modem control signals to remain active until after the data has cleared both the transmitter and the modem. This bit is always set in synchronous and SDLC modes.

### 5.3.3 Read Register 2

RR2 contains the interrupt vector written into WR2. When the register is accessed in Channel A, the vector returned is the vector actually stored in WR2. When this register is accessed in Channel B, the vector returned includes status information in bits 1, 2 and 3 or in bits 6, 5 and 4, depending on the state of the Status High/Status Low bit in WR9 and independent of the state of the VIS bit in WR9. The vector is modified according to Table 5-6 shown in the explanation of the VIS bit in WR9 (Section 5.2.11). If no interrupts are pending, the status is V3,V2,V1 -011, or V6,V5,V4-110. Figure 5-21 shows the bit positions for RR2.

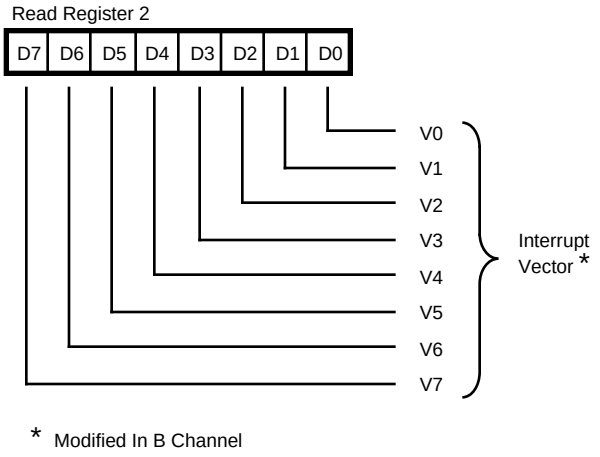



**Figure 5-21. Read Register 2**

### 5.3.4 Read Register 3

RR3 is the interrupt Pending register. The status of each of the interrupt Pending bits in the SCC is reported in this register. This register exists only in Channel A. If this register is accessed in Channel B, all 0s are returned. The two unused bits are always returned as 0. Figure 5-22 shows the bit positions for RR3.

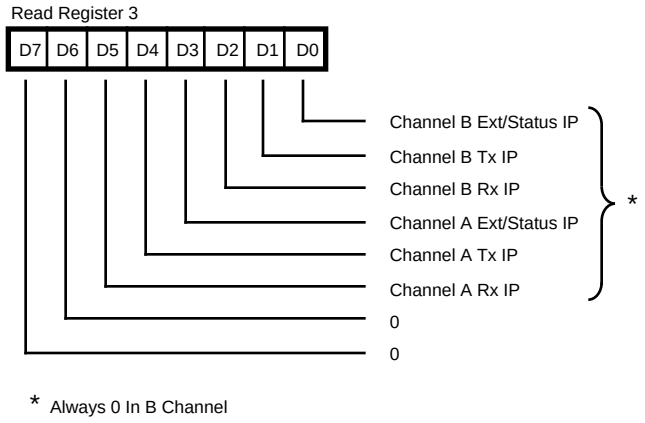



**Figure 5-22. Read Register 3**

### 5.3.5 Read Register 4 (ESCC and 85C30 Only)

On the ESCC, Read Register 4 reflects the contents of Write Register 4 provided the Extended Read option is enabled. Otherwise, this register returns an image of RR0.

On the NMOS/CMOS version, a read to this location returns an image of RR0.

### 5.3.6 Read Register 5 (ESCC and 85C30 Only)

On the ESCC, Read Register 5 reflects the contents of Write Register 5 provided the Extended Read option is enabled. Otherwise, this register returns an image of RR1.

On the NMOS/CMOS version, a read to this register returns an image of RR1.

### 5.3.7 Read Register 6 (Not on NMOS)

On the CMOS and ESCC, Read Register 6 contains the least significant byte of the frame byte count that is currently at the top of the Status FIFO. RR6 is shown in Figure 5-23. This register is readable only if the FIFO is enabled (refer to the description Write Register 15, bit D2 and Section 4.4.3). Otherwise, this register is an image of RR2.

On the NMOS version, a read to this register location returns an image of RR2.

### 5.3.8 Read Register 7 (Not on NMOS)

On the CMOS and ESCC, Read Register 7 contains the most significant six bits of the frame byte count that is currently at the top of the Status FIFO. Bit D7 is the FIFO Overflow Status and bit D6 is the FIFO Data Available Status. The status indications are given in Table 5-13. RR7 is shown in Figure 5-24. This register is readable only if the FIFO is enabled (refer to the description Write Register 15, bit D2). Otherwise this register is an image of RR3. Note, for proper operation of the FIFO and byte count logic, the registers should be read in the following order: RR7, RR6, RR1.

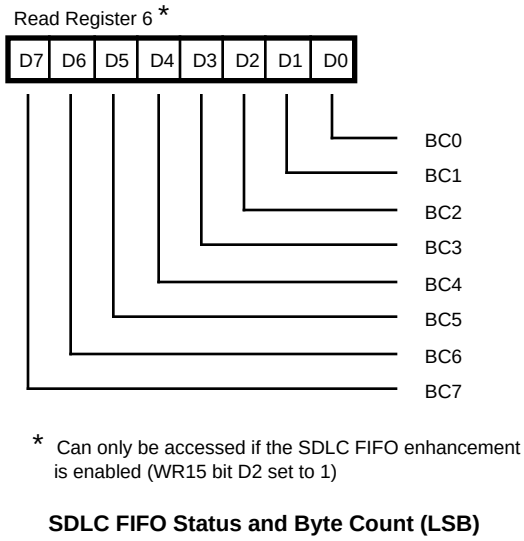
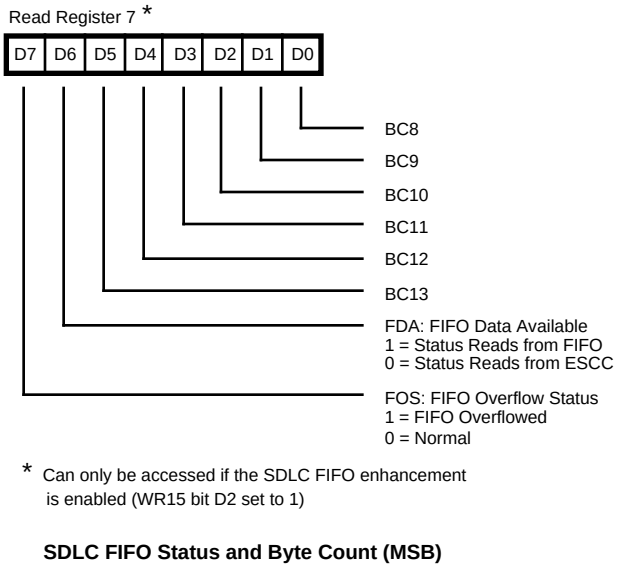



**Figure 5-23. Read Register 6 (Not on NMOS)**

## 5.3 READ REGISTERS (Continued)




* Can only be accessed if the SDLC FIFO enhancement is enabled (WR15 bit D2 set to 1)

**SDLC FIFO Status and Byte Count (MSB)**

**Figure 5-24. Read Register 7 (Not on NMOS)**

**Table 5-13. .Read Register 7 FIFO Status Decoding**

| Bit D7 | FIFO Data Available Status |
|---|---|
| 1 | Status reads come from FIFO (FIFO is not Empty) |
| 0 | Status reads bypass FIFO because FIFO is Empty) |
| **Bit D6** | **FIFO Overflow Status** |
| 1 | FIFO has overflowed |
| 0 | Normal operation |

If the FIFO overflows, the FIFO and the FIFO Overflow Status bit are cleared by disabling and then re-enabling the FIFO through the FIFO control bit (WR15, D2). Otherwise, this register returns an image of RR3.

On the NMOS version, a read to this location returns an image of RR3.

### 5.3.9 Read Register 8

RR8 is the Receive Data register.

### 5.3.10 Read Register 9 (ESCC and 85C30 Only)

On the ESCC, Read Register 9 reflects the contents of Write Register 3 provided the Extended Read option has been enabled.

On the NMOS/CMOS version, a read to this location returns an image of RR13.

### 5.3.11 Read Register 10

RR10 contains some miscellaneous status bits. Unused bits are always 0. Bit positions for RR10 are shown in Figure 5-25.

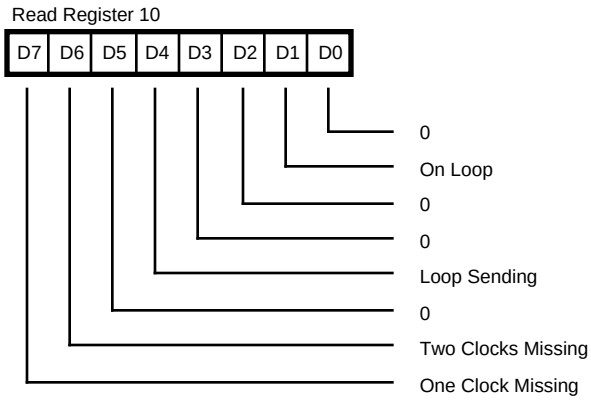



**Figure 5-25. Read Register 10**

**Bit 7: One Clock Missing status**
While operating in the FM mode, the DPLL sets this bit to 1 when it does not see a clock edge on the incoming lines in the window where it expects one. This bit is latched until reset by a Reset Missing Clock or Enter Search Mode command in WR14. In the NRZI mode of operation and while the DPLL is disabled, this bit is always 0.

**Bit 6: Two Clocks Missing status**
While operating in the FM mode, the DPLL sets this bit to 1 when it does not see a clock edge in two successive tries. At the same time the DPLL enters the Search mode. This bit is latched until reset by a Reset Missing Clock or Enter Search Mode command in WR14, bit 5-7. In the NRZI mode of operation and while the DPLL is disabled, this bit is always 0.

**Bit 4: Loop Sending status**
This bit is set to 1 in SDLC Loop mode while the transmitter is in control of the Loop, that is, while the SCC is actively transmitting on the loop. This bit is reset at all other times.

This bit can be polled in SDLC mode to determine when the closing flag has been sent.

**Bit 1: On Loop status**
This bit is set to 1 while the SCC is actually on loop in SDLC Loop mode. This bit is set to 1 in the X21 mode (Loop mode selected while in monosync) when the transmitter goes active. This bit is 0 at all other times. This bit can also be pulled in SDLC mode to determine when the closing flag has been sent.

### 5.3.12 Read Register 11 (ESCC and 85C30 Only)

On the ESCC, Read Register 11 reflects the contents of Write Register 10 provided the Extended Read option has been enabled. Otherwise, this register returns an image of RR15.

On the NMOS/CMOS version, a read to this location returns an image of RR15.

### 5.3.13 Read Register 12

RR12 returns the value stored in WR12, the lower byte of the time constant, for the BRG. Figure 5-26 shows the bit positions for RR12.

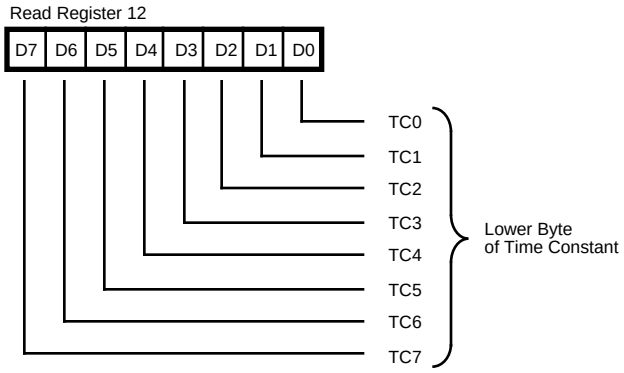



**Figure 5-26. Read Register 12**

### 5.3.14 Read Register 13

RR13 returns the value stored in WR13, the upper byte of the time constant for the BRG. Figure 5-27 shows the bit positions for RR13.

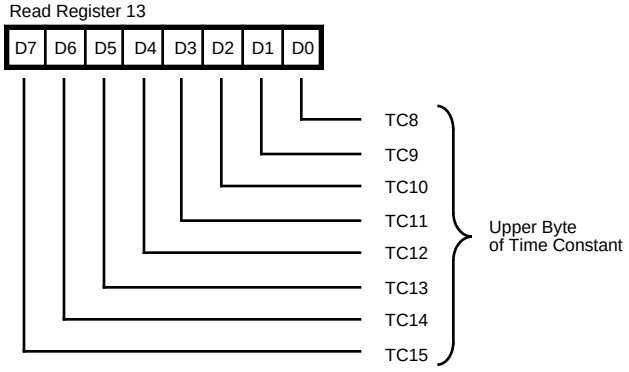



**Figure 5-27. Read Register 13**

### 5.3.15 Read Register 14 (ESCC and 85C30 Only)

On the ESCC, Read Register 14 reflects the contents of Write Register 7 Prime provided the Extended Read option has been enabled. Otherwise, this register returns an image of RR10.

On the NMOS/CMOS version, a read to this location returns an image of RR10.

### 5.3.16 Read Register 15

RR15 reflects the value stored in WR15, the External/Status IE bits. The two unused bits are always returned as Os. Figure 5-28 shows the bit positions for RR15.

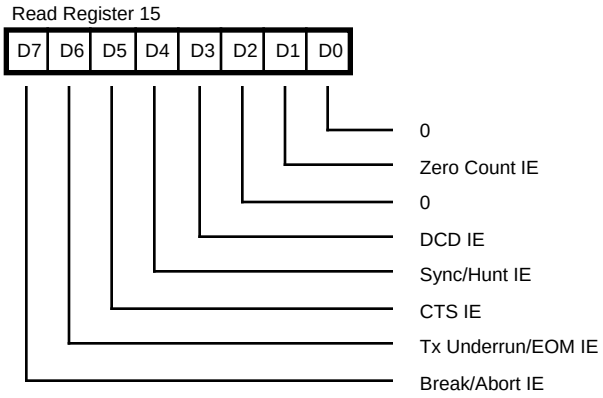



**Figure 5-28. Read Register 15**

# INTERFACING Z80® CPUS TO THE Z8500 PERIPHERAL FAMILY

## INTRODUCTION

The Z8500 Family consists of universal peripherals that can interface to a variety of microprocessor systems that use a non-multiplexed address and data bus. Though similar to Z80 peripherals, the Z8500 peripherals differ in the way they respond to I/O and Interrupt Acknowledge cycles. In addition, the advanced features of the Z8500 peripherals enhance system performance and reduce processor overhead.

To design an effective interface, the user needs an understanding of how the Z80 Family interrupt structure works, and how the Z8500 peripherals interact with this structure. This application note provides basic information on the interrupt structures, as well as a discussion of the hardware and software considerations involved in interfacing the Z8500 peripherals to the Z80 CPUs. Discussions center around each of the following situations:

- Z80A 4 MHz CPU to Z8500 4 MHz peripherals
- Z80B 6 MHz CPU to Z8500A 6 MHz peripherals
- Z80H 8 MHz CPU to Z8500 4 MHz peripherals
- Z80H 8 MHz CPU to Z8500A 6 MHz peripherals

This application note assumes the reader has a strong working knowledge of the Z8500 peripherals; it is not intended as a tutorial.

## CPU HARDWARE INTERFACING

The hardware interface consists of three basic groups of signals; data bus, system control, and interrupt control, described below. For more detailed signal information, refer to Zilog's DataBook, Universal Peripherals.

### Data Bus Signals

**D7-D0.** Data Bus (bidirectional tri-state). This bus transfers data between the CPU and the peripherals.

### System Control Signals

**AD-A0.** Address Select Lines (optional). These lines select the port and/or control registers.

**/CE.** Chip Enable (input, active Low). /CE is used to select the proper peripheral for programming. /CE should be gated with /IORQ or /MREQ to prevent spurious chip selects during other machine cycles.

**/RD*** Read (input, active Low). /RD activates the chip-read circuitry and gates data from the chip onto the data bus.

**/WR*** Write (input, active Low). /WR strobes data from the data bus into the peripheral.

*Chip reset occurs when /RD and /WR are active simultaneously.

### Interrupt Control

**/INTACK.** Interrupt Acknowledge (input, active Low). This signal indicates an Interrupt Acknowledge cycle and is used with /RD to gate the interrupt vector onto the data bus.

**/INT**. Interrupt Request (output, open-drain, active Low).

The IUS bit indicates that an interrupt is currently being serviced by the CPU. The IUS bit is set during an Interrupt Acknowledge cycle if the IP bit is set and the IEI line is High. If the IEI line is Low, the IUS bit is not set, and the device is inhibited from placing its vector onto the data bus. In the Z80 peripherals, the IUS bit is normally cleared by decoding the RETI instruction, but can also be cleared by a software command (SIO). In the Z8500 peripherals, the IUS bit is cleared only by software commands.

## CPU HARDWARE INTERFACING (Continued)

### Z80® Interrupt Daisy-Chain Operation

In the Z80 peripherals, both the IP and IUS bits control the IEO line and the lower portion of the daisy chain.

When a peripheral's IP bit is set, its IEO line is forced Low. This is true regardless of the state of the IEI line. Additionally, if the peripheral's IUS bit is clear and its IEI line High, the /INT line is also forced Low.

The Z80 peripherals sample for both /M1 and /IORQ active, and /RD inactive to identify and Interrupt Acknowledge cycle. When /M1 goes active and /RD is inactive, the peripheral detects an Interrupt Acknowledge cycle and allows its interrupt daisy chain to settle. When the /IORQ line goes active with /M1 active, the highest priority interrupting peripheral places its interrupt vector onto the data bus. The IUS bit is also set to indicate that the peripheral is currently under service. As long as the IUS bit is set, the IEO line is forced Low. This inhibits any lower priority devices from requesting an interrupt. When the Z80 CPU executes the RETI instruction, the peripherals monitor the data bus and the highest priority device under service resets its IUS bit.

### Z8500 Interrupt Daisy-Chain Operation

In the Z8500 peripherals, the IUS bit normally controls the state of the IEO line. The IP bit affects the daisy chain only during an Interrupt Acknowledge cycle. Since the IP bit is normally not part of the Z8500 peripheral interrupt daisy chain, there is no need to decode the RETI instruction. To allow for control over the daisy chain, Z8500 peripherals have a Disable Lower Chain (DLC) software command that pulls IEO Low. This can be used to selectively deactivate parts of the daisy chain regardless of the interrupt status. Table 1 shows the truth tables for the Z8500 interrupt daisy-chain control signals during certain cycles. Table 2 shows the interrupt state diagram for the Z8500 peripherals.

**Table 1. Z8500 Daisy-Chain Control Signals**

| Truth Table for Daisy Chain Signals During Idle State | | | | Truth Table for Daisy Chain Signals During /INTACK Cycle | | | |
|---|---|---|---|---|---|---|---|
| **IEI** | **IP** | **IUS** | **IEO** | **IEI** | **IP** | **IUS** | **IEO** |
| 0 | X | X | 0 | 0 | X | X | 0 |
| 1 | X | 0 | 1 | 1 | 1 | X | 0 |
| 1 | X | 1 | 0 | 1 | X | 1 | 0 |
| 1 | 0 | 0 | 1 | | | | |

**IEI.** Interrupt Enable In (Input, active High).

**IEO.** Interrupt Enable Out (output, active High).

These lines control the interrupt daisy chain for the peripheral interrupt response.

### Z8500 I/O Operation

The Z8500 peripherals generate internal control signals from /RD and /WR. Since PCLK has not required phase relationship to /RD or /WR, the circuitry generating these signals provides time for metastable conditions to disappear.

The Z8500 peripherals are initialized for different operating modes by programming the internal registers. These internal registers are accessed during I/O Read and Write cycles, which are described below.

### Read Cycle Timing

Figure 1 illustrates the Z8500 Read cycle timing. All register addresses and /INTACK must remain stable throughout the cycle. If /CE goes active after /RD goes active, or if /CE goes inactive before /RD goes inactive, then the effective Read cycle is shortened.

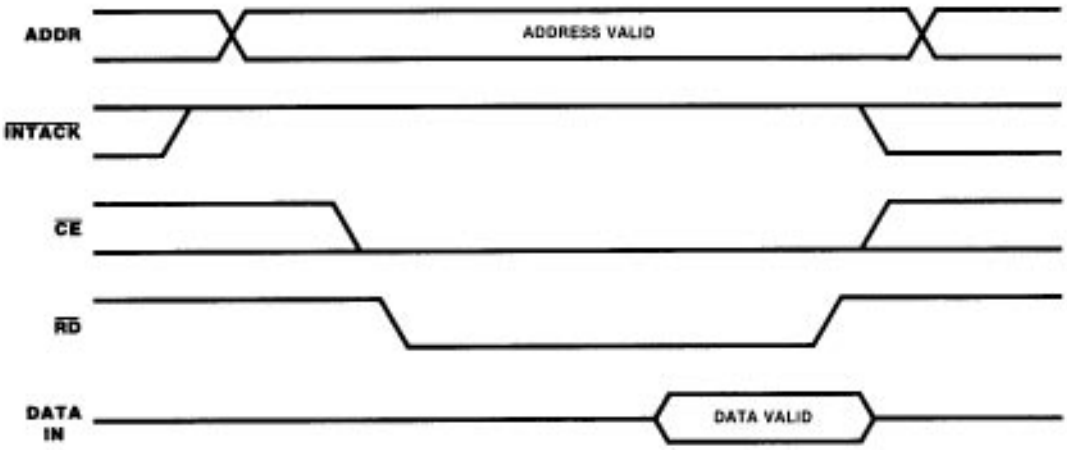



**Figure 1. Z8500 Peripheral I/O Read Cycle Timing**

## Write Cycle Timing

Figure 2 illustrates the Z8500 Write cycle timing. All register addresses and /INTACK must remain stable throughout the cycle. If /CE goes active after /WR goes active, or if /CE goes inactive before /WR goes inactive, then the effective Write cycle is shortened. Data must be available to the peripheral prior to the falling edge of /WR.

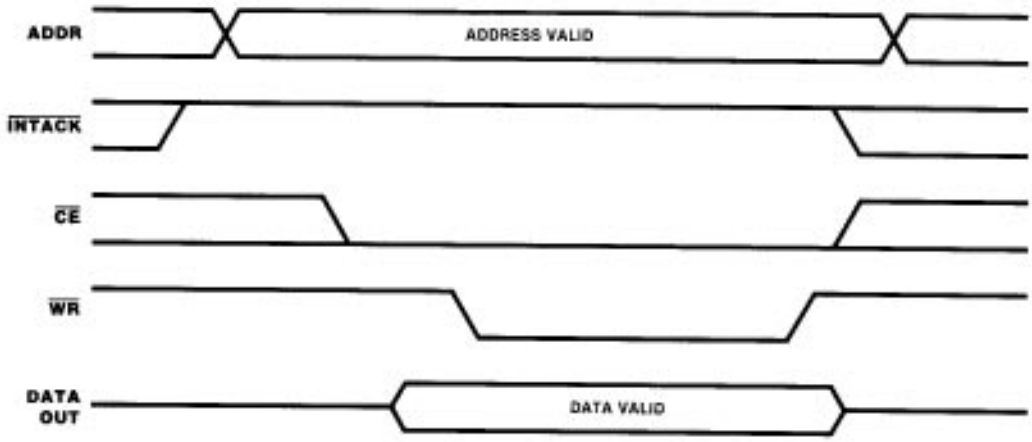



**Figure 2. Z8500 Peripheral I/O Write Cycle Timing**

## PERIPHERAL INTERRUPT OPERATION

Understanding peripheral interrupt operation requires a basic knowledge of the Interrupt Pending (IP) and Interrupt Under Service (IUS) bits in relation to the daisy chain. Both Z80 and Z8500 peripherals are designed in such a way that no additional interrupts can be requested during an Interrupt Acknowledge cycle. This allows that interrupt daisy chain to settle, and ensures proper response of the interrupting device.

The IP bit is set in the peripheral when CPU intervention is required (such conditions as buffer empty, character available, error detection, or status changes). The Interrupt Acknowledge cycle does not necessarily reset the IP bit. This bit is cleared by a software command to the peripheral, or when the action that generated the interrupt is completed (i.e., reading a character, writing data, resetting errors, or changing the status). When the interrupt has been serviced, other interrupts can occur.

The Z8500 peripherals use /INTACK (Interrupt Acknowledge) for recognition of an Interrupt Acknowledge cycle. This pin, used in conjunction with /RD, allows the Z8500 peripheral to gate its interrupt vector onto the data bus. An active /RD signal during an Interrupt Acknowledge cycle performs two functions. First, it allows the highest priority device requesting an interrupt to place its interrupt vector on the data bus. Secondly, it sets the IUS bit in the highest priority device to indicate that the device is currently under service.

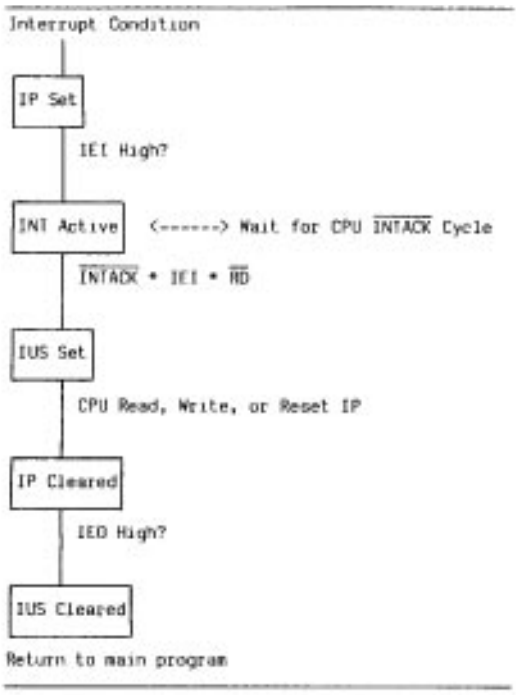



**Figure 3. Z8500 Interrupt State Diagram**

## INPUT/OUTPUT CYCLES

Although Z8500 peripherals are designed to be as universal as possible, certain timing parameters differ from the standard Z80 timing. The following sections discuss the I/O interface for each of the Z80 CPUs and the Z8500 peripherals. Figure 9 depicts logic for the Z80A CPU to Z8500 peripherals (and Z80B CPU to Z8500A peripherals) I/O interface as well as the Interrupt Acknowledge interface. Figures 4 and 7 depict some of the logic used to interface the Z80H CPU to the Z8500 and Z8500A peripherals for the I/O and Interrupt Acknowledge interfaces. The logic required for adding additional Wait states into the timing flow is not discussed in the following sections.

### Z80A CPU to Z8500 Peripherals

No additional Wait states are necessary during the I/O cycles, although additional Wait states can be inserted to compensate for timing delays that are inherent in a system. Although the Z80A timing parameters indicate a negative value for data valid prior to /WR, this is a worse than “worst case” value. This parameter is based upon the longest (worst case) delay for data available from the falling edge of the CPU clock minus the shortest (best case) delay for CPU clock High to /WR low. The negative value resulting from these two parameters does not occur because the worst case of one parameter and the best case of the other do not occur within the same device. This indicates that the value for data available prior to /WR will always be greater than zero.

All setup and pulse width times for the Z8500 peripherals are met by the standard Z80A timing. In determining the interface necessary, the /CE signal to the Z8500 peripherals is assumed to be the decoded address qualified with the /IORQ signal.

Figure 4 shows the minimum Z80A CPU to Z8500 peripheral interface timing for I/O cycles. If additional Wait states are needed, the same number of Wait states can be inserted for both I/O Read and Write cycles to simplify interface logic. There are several ways to place the Z80A CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500 I/O cycles. Tables 3 and 4 list the Z8500 peripheral and the Z80A CPU timing parameters (respectively) of concern during the I/O cycles. Tables 5 and 6 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 3 and 4 refer to the timing diagram in Figure 4.

## INPUT/OUTPUT CYCLES (Continued)

**Table 2. Z8500 Timing Parameters I/O Cycles**

| Worst Case | | | Min | Max | Units |
|---|---|---|---|---|---|
| 6. | TsA(WR) | Address to /WR to Low Setup | 80 | | ns |
| 1. | TsA(RD) | Address to /RD Low Setup | 80 | | ns |
| 2. | TdA(DR) | Address to Read Data Valid | | 590 | |
| | TsCEI(WR) | /CE Low to /WR Low Setup | | ns | |
| | TsCEI(RD) | /CE Low to /RD Low Setup | | ns | |
| 4. | TwRDI | /RD Low Width | 390 | | ns |
| 8. | TwWRI | /WR Low Width | 390 | | ns |
| 3. | TdRDf(DR) | /RD Low to Read Data Valid | | 255 | ns |
| 7. | TsDW(WR) | Write Data to /WR Low Setup | 0 | | ns |

**Table 3. Z80A Timing Parameters I/O Cycles**

| Worst Case | | | Min | Max | Units |
|---|---|---|---|---|---|
| | TcC | Clock Cycle Period | 250 | | ns |
| | TwCh | Clock Cycle High Width | 110 | | ns |
| | TfC | Clock Cycle Fall Time | | 30 | ns |
| | TdCr(A) | Clock High to Address Valid | | 110 | ns |
| | TdCr(RDf) | Clock High to /RD Low | | 85 | ns |
| | TdCr(IORQf) | Clock High to /IORQ Low | | 75 | ns |
| | TdCr(WRf) | Clock High to /WR Low | | 65 | ns |
| 5. | TsD(Cf) | Data to Clock Low Setup | 50 | | ns |

**Table 4. Parameter Equations**

| Z8500 Parameter | Z80A Equation | Value | Units |
|---|---|---|---|
| TsA(RD) | TcC-TdCr(A) | 140 min | ns |
| TdA(DR) | 3TcC+TwCh-TdCr(A)-TsD(Cf) | 800 min | ns |
| TdRDf(DR) | 2TcC+TwCh-TsD(Cf) | 460 min | ns |
| TwRD1 | 2TcC+TwCh+TfC-TdCr(RDf) | 525 min | ns |
| TsA(WR) | TcC-TdCr (A) | 140 min | ns |
| TsDW(WR) | | >0 min | ns |
| TwWR1 | 2TcC+TwCh+TfC-TdCr(WRf) | 560 min | ns |

.

**Table 5. Parameter Equations**

| Z80A Parameter | Z8500 Equation | Value | Units |
|---|---|---|---|
| TsD(Cf) | 3TcC+TwCh-TdCr(A)-TdA(DR) | 160 min | ns |
| | /RD | | |
| | 2TcC+TwCh-TdCr(RDf)-TdRD(DR) | 135 min | ns |

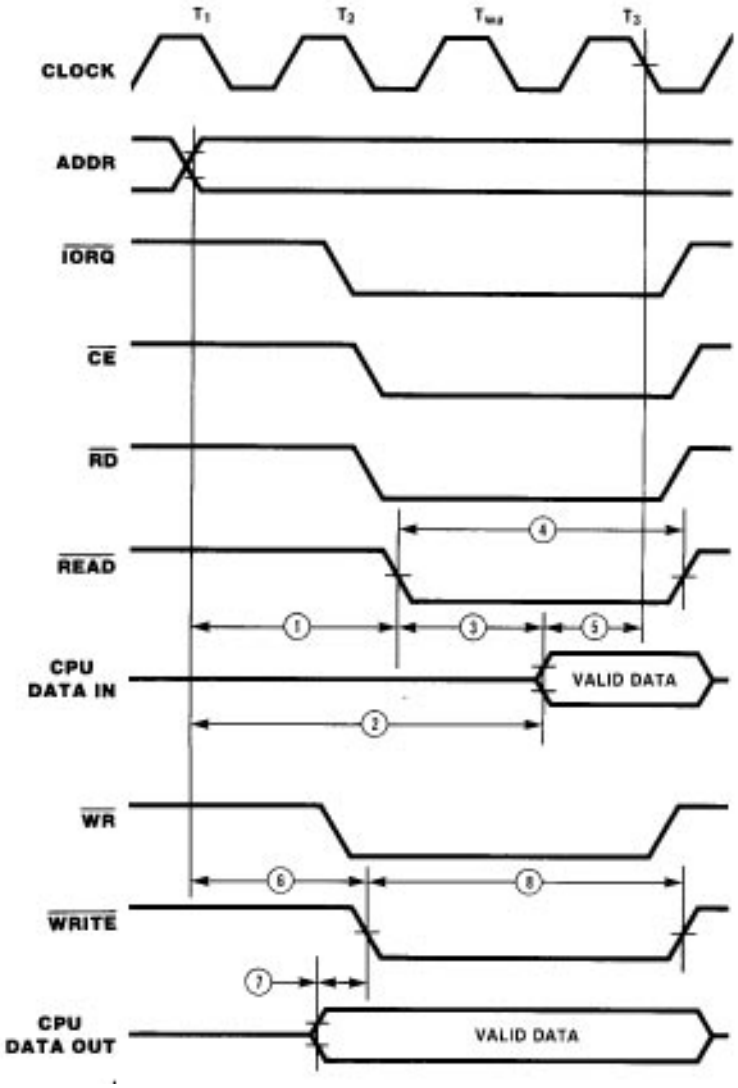



**Figure 4. Z80A CPU to Z8500 Peripheral Minimum I/O Cycle Timing**

## Z80B CPU TO Z8500A PERIPHERALS

No additional Wait states are necessary during I/O cycles, although Wait states can be inserted to compensate for any systems delays. Although the Z80B timing parameters indicate a negative value for data valid prior to /WR, this is a worse than “worst case” value. This parameter is based upon the longest (worst case) delay for data available from the falling edge of the CPU clock minus the shortest (best case) delay for CPU clock High to /WR Low. The negative value resulting from these two parameters does not occur because the worst case of one parameter and best case of the other do not occur within the same device. This indicates that the value for data available prior to /WR will always be greater than zero.

All setup and pulse width times for the Z8500A peripherals are met by the standard Z80B timing. In determining the interface necessary, the /CE signal to the Z8500A peripherals is assumed to be the decoded address qualified with /IORQ signal.

Figure 5 shows the minimum Z80B CPU to Z8500A peripheral interface timing for I/O cycles. If additional Wait states are needed, the same number of Wait states can be inserted for both I/O Read and I/O Write cycles in order to simplify interface logic. There are several ways to place the Z80B CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500A I/O cycles. Tables 6 and 7 list the Z8500A peripheral and Z80B CPU timing parameters (respectively) of concern during the I/O cycles. Tables 8 and 9 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 6 and 7 refer to the timing diagram of Figure 5.

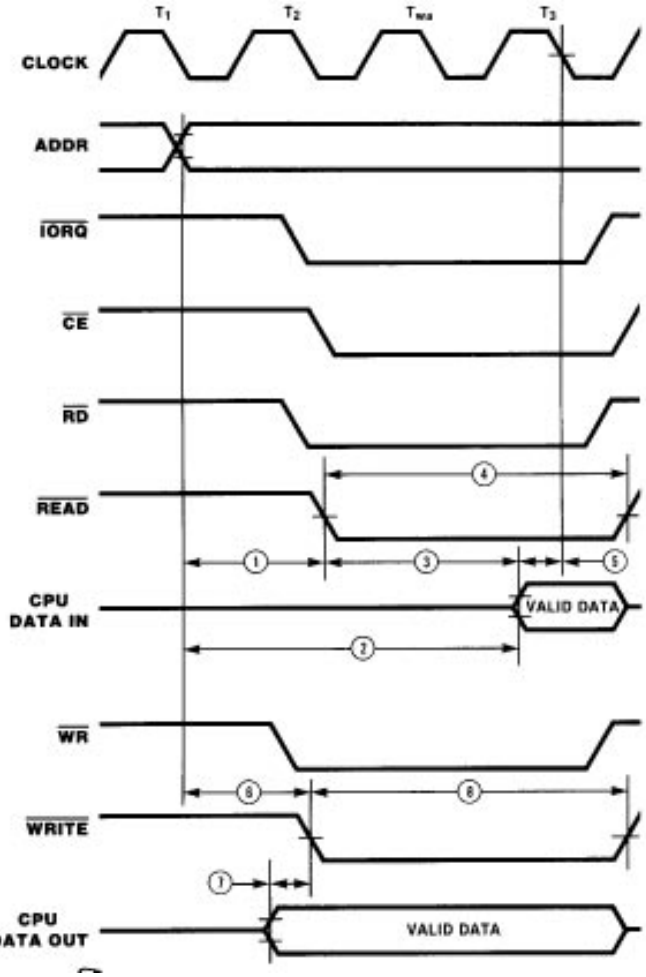



**Figure 5. Z80B CPU to Z8500A Peripheral Minimum I/O Cycle Timing**

**Table 6. Z8500A Timing Parameters I/O Cycles**

| Worst Case | | | Min | Max | Units |
|---|---|---|---|---|---|
| 6. | TsA(WR) | Address to /WR Low Setup | 80 | | ns |
| 1. | TsA(RD) | Address to /RD Low Setup | 80 | | ns |
| 2. | TdA(DR) | Address to Read Data Valid | | 420 | ns |
| | TsCE1(WR) | /CE Low to /WR Low Setup | | ns | |
| | TsCE1(RD) | /CE Low to /RD Low Setup | | ns | |
| 4. | TwRD1 | /RD Low Width | 250 | | ns |
| 8. | TwWR1 | /WR Low Width | 250 | | ns |
| 3. | TdRDf(DR) | /RD Low to Read Data Valid | | 180 | ns |
| 7. | TsDW(WR) | Write Data to /WR Low Setup | 0 | | ns |

**Table 7. Z80B Timing Parameters I/O Cycles**

| Worst Case | | | Min | Max | Units |
|---|---|---|---|---|---|
| | TcC | Clock Cycle Period | 165 | | ns |
| | TwCh | Clock Cycle High Width | 65 | | ns |
| | TfC | Clock Cycle Fall Time | | 20 | ns |
| | TdCr(A) | Clock High to Address Valid | | 90 | ns |
| | TdCr(RDf) | Clock High to /RD Low | | 70 | ns |
| | TdCR(IORQf) | Clock High to /IORQ Low | | 65 | ns |
| | TdCr(WRf) | Clock High to /WR Low | | 60 | ns |
| 5. | TsD(Cf) | Data to Clock Low Setup | 40 | | ns |

**Table 8. Parameter Equations**

| Z8500A Parameter | Z80B Equation | Value | Units |
|---|---|---|---|
| TsA(RD) | TcC-TdCr(A) | >75 min | ns |
| TdA(DR) | 3TcC+TwCh-TdCr(A)-TsD(Cf) | 430 min | ns |
| TdRDf(DR) | 2TcC+TwCh+TsD(Cf) | 345 min | ns |
| TwRD1 | 2TcC+TwCh+TfC-TdCr(RDf) | 325 min | ns |
| TsA(WR) | TcC-TdCr(A) | 75 min | ns |
| TsDW(WR) | | > 0 min | ns |
| TwWR1 | 2 TcC+Twch+TfC-TdCr(WRf) | 352 min | ns |

**Table 9. Parameter Equations**

| Z8500A Equation | | Value | Units |
|---|---|---|---|
| | 3TcC+TwCh-TdCr(A)-TdA(DR) | 50 min | ns |
| | 2TcC+TwCh-TdCr(RDf)-TdRD(DR) | 75 min | ns |
| | Z80H CPU to Z8500 Peripherals | | |

## Z90H CPU TO Z8500 PERIPHERALS

During an I/O Read cycle, there are three Z8500 parameters that must be satisfied. Depending upon the loading characteristics of the /RD signal, the designer may need to delay the leading (falling) edge of /RD to satisfy the Z8500 timing parameter TsA(RD) (Addresses Valid to /RD Setup). Since Z80H timing parameters indicate that the /RD signal may go Low after the falling edge of T2, it is recommended that the rising edge of the system clock be used to delay /RD (if necessary). The CPU must also be placed into a Wait condition long enough to satisfy TdA(DR) (Address Valid to Read Data Valid Delay) and TdRDf(DR) (/RD Low to Read Data Valid Delay).

During an I/O Write cycle, there are three other Z8500 parameters that must be satisfied. Depending upon the loading characteristics of the /WR signal and the data bus, the designer may need to delay the leading (falling) edge of /WR to satisfy the Z8500 timing parameters TsA(WR) (Address Valid to /WR setup). Since Z80H timing parameters indicate that the /WR signal may go Low after the falling edge of T2, it is recommended that the rising edge of the system clock be used to delay /WR (if necessary). This delay will ensure that both parameters are satisfied. The CPU must also be placed into a Wait condition long enough to satisfy TwWR1 (/WR Low Pulse Width). Assuming that the /WR signal is delayed, only two additional Wait states are needed during an I/O Write cycle when interfacing the Z80H CPU to the Z8500 peripherals.

To simplify the I/O interface, the designer can use the same number of Wait states for both I/O Read and I/O Write cycles. Figure 6 shows the minimum Z80H CPU to Z8500 peripheral interface timing for the I/O cycles (assuming that the same number of Wait states are used for both cycles and that both /RD and /WR need to be delayed). Figure 8 shows two suits that can be used to delay the leading (falling) edge of either the /RD or the /WR signals. There are several ways to place the Z80A CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the use wants to place Wait states in all I/O cycles, or only during Z8500 I/O cycles. Tables 3 and 10 list the Z8500 peripheral and the Z80H CPU timing parameters (respectively) of concern during the I/O cycles. Tables 13 and 14 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Tables 3 and 10 refer to the timing diagram of Figure 6.

**Table 10. Z80H Timing Parameter I/O Cycles**

| | | Equation | Min | Max | Units |
|---|---|---|---|---|---|
| | TcC | Clock Cycle Period | 125 | | |
| | TwCh | Clock Cycle High Width | 55 | | ns |
| | TfC | Clock Cycle Fall Time | | 10 | ns |
| | TdCr(A) | Clock High to Address Valid | | 80 | ns |
| | TdCr(RDf) | Clock High to /RD Low | | 60 | ns |
| | TdCr(IORQf) | Clock High to /IORQ Low | | 55 | ns |
| | TdCr(WRf) | Clock High to /WR Low | | 55 | ns |
| 5. | TsD(Cf) | Data to Clock Low Setup | 30 | | ns |

**Table 11. Parameter Equations**

| Z8500 Parameter | Z80H Equation | Value | Units |
|---|---|---|---|
| TsA(RD) | 2TcC-TdCr(A) | 170 min | ns |
| TdA(DR) | 6TcC+TwCh-TdCr(A)-TsD(Cf) | 695 min | ns |
| TdRDf(DR) | 4TcC+TwCh-TsD(Cf) | 523 min | ns |
| TwRD1 | 4TcC+TwCh+TfC-TdCr(RDf) | 503 min | ns |
| TsA(WR) | /WR - delayed | | |
| | 2TcC-TdCr(A) | 170 min | ns |
| TsDW(WR) | | >0 min | ns |
| TwWR1 | 4TcC+TwCh+TfC | 563 min | ns |

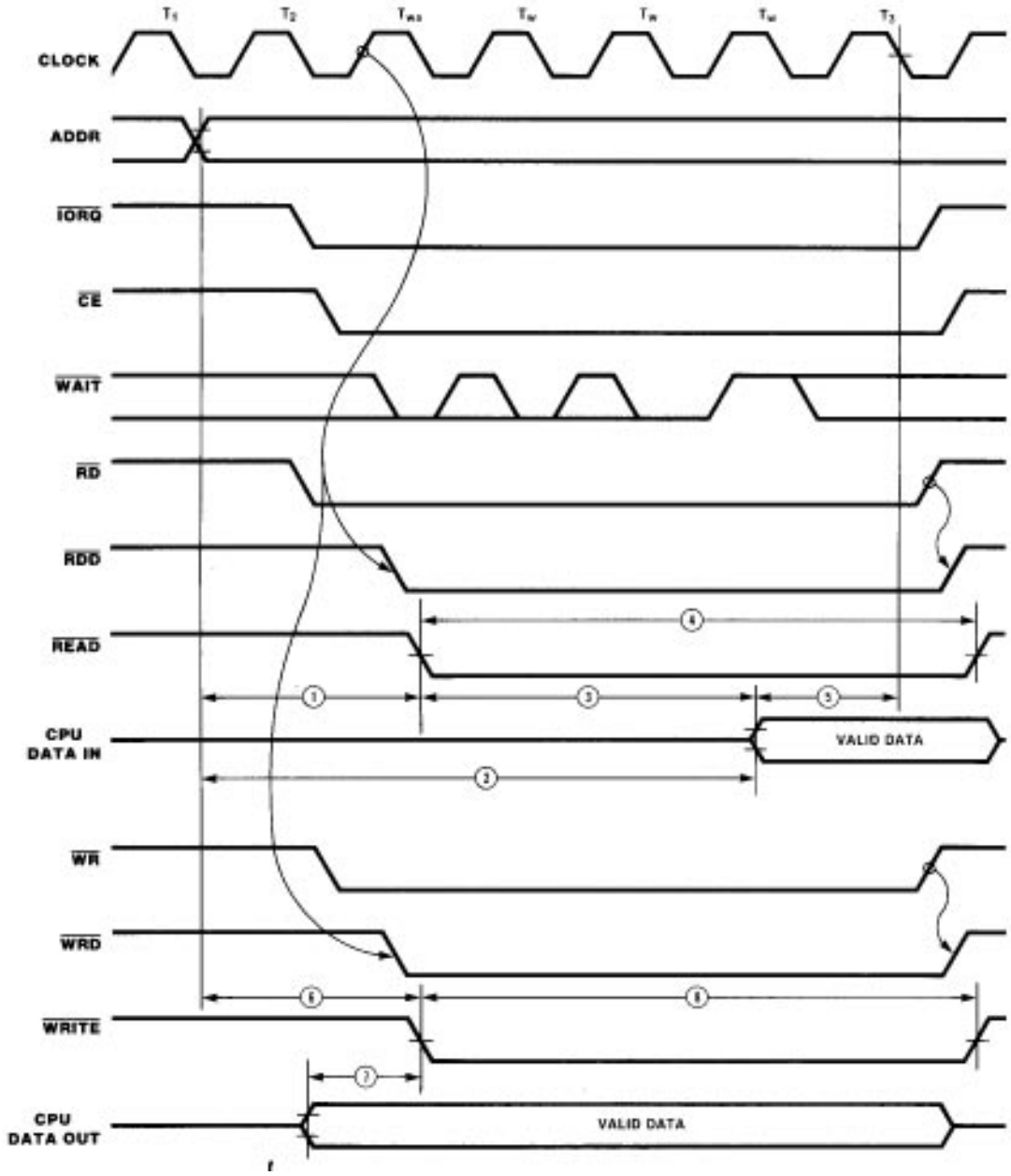



**Figure 6. Z80H CPU to Z8500 Peripheral Minimum I/O Cycle Timing**

## Z80H CPU TO Z8500A PERIPHERALS

During an I/O Read cycle, there are three Z8500A parameters that must be satisfied. Depending upon the loading characteristics of the /RD signal, the designer may need to delay the leading (falling) edge of /RD to satisfy the Z8500A timing parameter TsA(RD) (Address Valid to /RD Setup). Since Z80H timing parameters indicate that the /RD signal may go Low after the falling edge of T2, it is recommended that the rising edge of the system must also be placed into Wait condition long enough to satisfy TdA(DR) (Address Valid to Read Data Valid Delay) and TdRDf(DR) (/RD Low to Read Data Valid Delay). Assuming that the /RD signal is delayed, then only one additional Wait state is needed during an I/O Read cycle when interfacing the Z80H CPU to the Z8500A peripherals.

During an I/O Write cycle, there are three other Z850A parameters that have to be satisfied. Depending upon the loading characteristics of the /WR signal and the data bus, the designer may need to delay the leading (falling) edge of /WR to satisfy the Z8500A timing parameters TsA(WR) (Address Valid to /WR Setup) and TsDW(WR) (Data Valid Prior to /WR Setup). Since Z80H timing parameters indicate that the /WR signal may go Low after the falling edge of T2, it is recommended that the rising edge of the system clock be used to delay /WR (if necessary). This delay will ensure that both parameters are satisfied. The CPU must also be placed into a Wait condition long enough to satisfy TwWR1 (/WR Low Pulse Width). Assuming that the /WR signal is delayed, then only one additional Wait state is needed during an I/O Write cycle when interfacing the Z80H CPU to the Z8500A peripherals.

Figure 7 shows the minimum Z80H CPU to Z8500A peripheral interface timing for the I/O cycles (assuming that the same number of Wait states are used for both cycles and that both /RD and /WR need to be delayed). Figure 8 shows two circuits that may be used to delay leading (falling) edge of either the /RD or the /WR signals. There are several methods used to place the Z80A CPU into a Wait condition (such as counters or shift registers to count system clock pulses), depending upon whether or not the user wants to place Wait states in all I/O cycles, or only during Z8500A I/O cycles, Tables 7 and 11 list the Z8500A peripheral and the Z80H CPU timing parameters (respectively) of concern during the I/O cycles. Tables 14 and 15 list the equations used in determining if these parameters are satisfied. In generating these equations and the values obtained from them, the required number of Wait states was taken into account. The reference numbers in Table 4 and 11 refer to the timing diagram of Figure 7.

**Table 12. Parameter Equations**

| Z80H Parameter | Z8500 Equation | Value | Units |
|---|---|---|---|
| TsD(Cf) | 6TcC+TwCh-TdCr(A)-TdA(DR) | 135 min | ns |
| | /RD - delayed | | |
| | 4TcC+TwCh+TfC-TdRD(DR) | 300 min | ns |

**Table 13. Parameter Equations**

| Z8500A Parameter | Z80H Equation | Value | Units |
|---|---|---|---|
| TsA(RD) | 2TcC-TdCr(A) | 170 min | ns |
| TdA(DR) | 6TcC+TwCh-TdCr(A)-TsD(Cf) | 695 min | ns |
| TdRDf(DR) | 4TcC+TwCh-TsD(Cf) | 525 min | ns |
| TwRD1 | 4TcC+TwCh+TfC-TdCr(RDf) | 503 min | ns |
| TsA(WR) | /WR - delayed | | |
| | 2TcC-TdCr(A) | 170 min | ns |
| TsDW(WR) | | >0 min | ns |
| TwWR1 | 2TcC+TwCh+TfC | 313 min | ns |

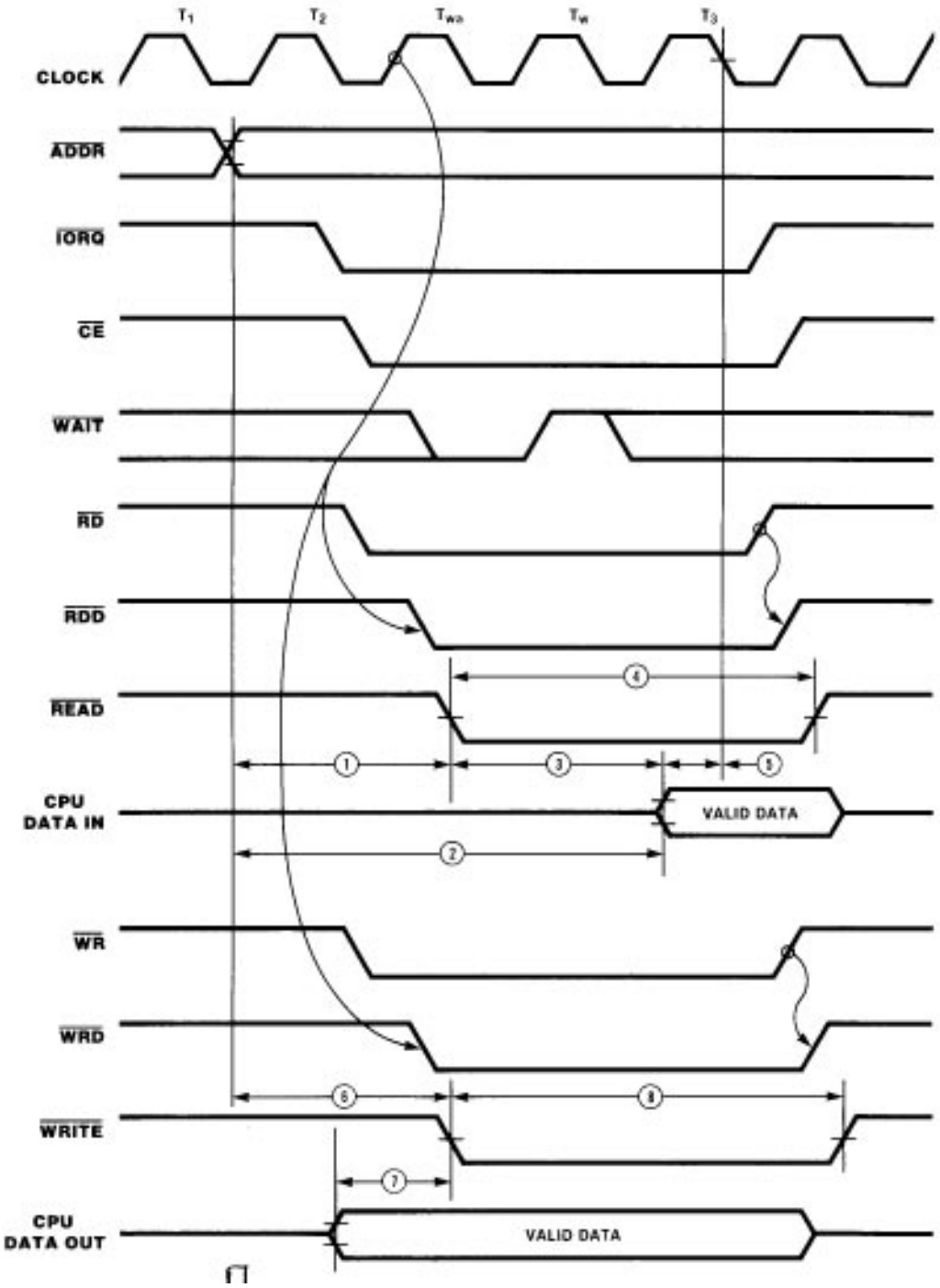



**Figure 7. Z80H CPU to Z8500A Peripheral Minimum I/O Cycle Timing**

## Z80H CPU TO Z8500A PERIPHERALS (Continued)

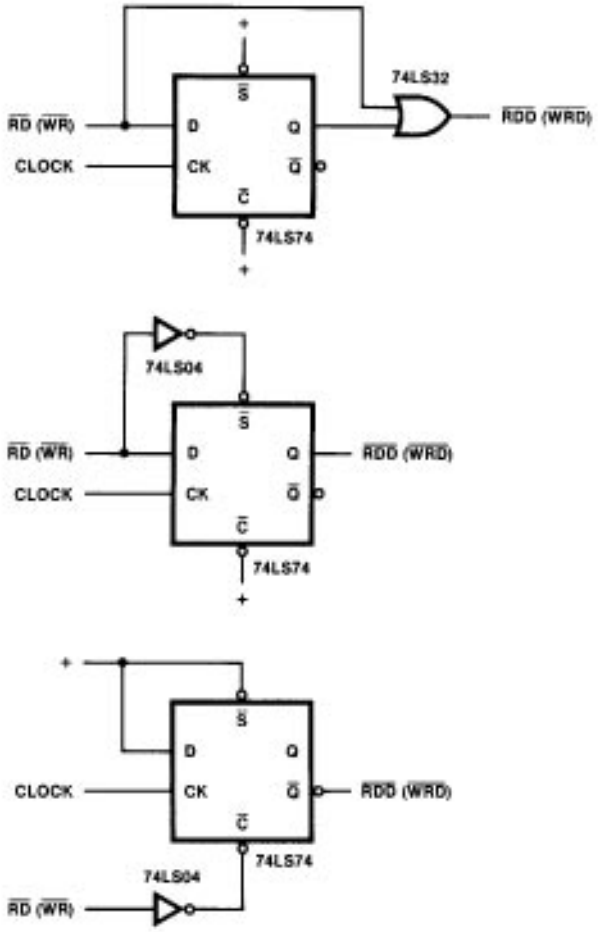



**Figure 8. Delaying /RD or /WR**

**Table 14. Parameter Equations**

| Z80H Parameter | Z8500A Equation | Value | Units |
|---|---|---|---|
| TsD(Cf) | 4TcC+TwCh-TdCr(A)-TdA(DR) /RS - delayed | 55 min | ns |
| | 2TcC+TwCh-TdRD(DR) | 125 min | ns |

## INTERRUPT ACKNOWLEDGE CYCLES

The primary timing differences between the Z80 CPUs and Z8500 peripherals occur in the Interrupt Acknowledge cycle. The Z8500 timing parameters that are significant during Interrupt Acknowledge cycles are listed in Table 16, while the Z80 parameters are listed in Table 17. The reference numbers in Tables 16 and 17 refer to Figures 10, 12 and 13.

If the CPU and the peripherals are running at different speeds (as with the Z80H interface), the /INTACK signal must be synchronized to the peripheral clock. Synchronization is discussed in detail under Interrupt Acknowledge for Z80H CPU to Z8500/8500A Peripherals.

During an Interrupt Acknowledge cycle, Z8500 peripherals require both /INTACK and /RD to be active at certain times. Since the Z80 CPUs do not issue either /INTACK or /RD, external logic must generate these signals.

Generating these two signals is easily accomplished, but the Z80 CPU must be placed into a Wait condition until the peripheral interrupt vector is valid. If more peripherals are added to the daisy chain, additional Wait states may be necessary to give the daisy chain time to settle. Sufficient time between /INTACK active and /RD active should be allowed for the entire daisy chain to settle.

Since the Z8500 peripheral daisy chain does not use the IP flag except during interrupt acknowledge, there is no need for decoding the RETI instruction used by the Z80 peripherals. In each of the Z8500 peripherals, there are commands that reset the individual IUS flags.

## EXTERNAL INTERFACE LOGIC

The following sections discuss external interface logic required during Interrupt Acknowledge cycles for each interface type.

### CPU/Peripheral Same Speed

Figure 9 shows the logic used to interface the Z80A CPU to the Z8500 peripherals and the Z80B CPU to Z8500A peripherals during an Interrupt Acknowledge cycle. The primary component in this logic is the Shift register (74LS164), which generates /INTACK, /READ, and /WAIT.

**Table 15. Z8500 Timing Parameters Interrupt Acknowledge Cycles**

| | | | 4 MHz | | 6 MHz | | |
|---|---|---|---|---|---|---|---|
| **Worst Case** | | | **Min** | **Max** | **Min** | **Max** | **Units** |
| 1. | TsIA(PC) | /INTACK Low to PCLK High Setup | 100 | | 100 | | ns |
| | ThIA(PC) | /INTACK Low to PCLK High Hold | 100 | | 100 | | ns |
| 2. | TdIAi(RD) | /INTACK Low to RD (Acknowledge) Low | 350 | | 250 | | ns |
| 5. | TwRDA | /RD (Acknowledge) Width | 350 | | 250 | | ns |
| 3. | TdRDA(DR) | /RD (Acknowledge) to Data Valid | | 250 | | 180 | ns |
| | TsIEI(RDA) | IEI to /RD (Acknowledge) Setup | 120 | | 100 | | ns |
| | ThIEI(RDA) | IEI to /RD (Acknowledge) Hold | 100 | | 70 | | ns |
| | TdIEI(IE) | IEI to IEO Delay | | 150 | | 100 | ns |

.

**Table 16. Z80 CPU Timing Parameters Interrupt Acknowledge Cycles**

| | | 4 MHz | | 6 MHz | | 8 MHz | | |
|---|---|---|---|---|---|---|---|---|
| **Worst Case** | | **Min** | **Max** | **Min** | **Max** | **Min** | **Max** | **Units** |
| TdC(M1f) | Clock High to /M1 Low Delay | | 100 | | 80 | | 70 | ns |
| TdM1f(IORQf) | /M1 Low to /IORQ Low Delay | 575* | | *345 | | 275* | | ns |
| 4. TsD(Cr) | Data to Clock High Setup | 35 | | 30 | | 25 | | ns |
| | *Z80A: 2TcC + TwCh + TfC - 65<br>Z80B: 2 TcC + TwCh + TfC - 50<br>Z80H: 2TcC + TwCh + TfC - 45 | | | | | | | |

## EXTERNAL INTERFACE LOGIC (Continued)

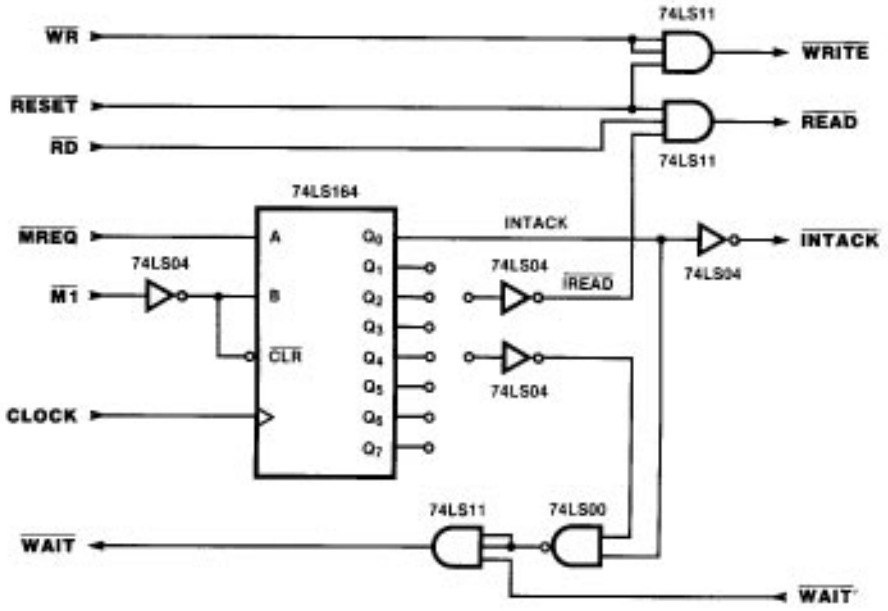



**Figure 9. Z80A/Z80B CPU to Z8500/Z8500A Peripheral Interrupt Acknowledge Interface Logic**

During I/O and normal memory access cycles, the Shift registers remains cleared because the /M1 signal is inactive. During opcode fetch cycles, also, the Shift register remains cleared, because only 0s can be clocked through the register. Since Shift register outputs are Low, /READ, /WRITE, and /WAIT are controlled by other system logic and gated through the AND gates (74LS11). During I/O and normal memory access cycles, /READ and /WRITE are active as a result of the system /RD and /WR signals (respectively) becoming active. If system logic requires that the CPU be placed into a Wait condition, the /WAIT signal controls the CPU. Should it be necessary to reset the system, /RESET causes the interface logic to generate both /READ and /WRITE (the Z8500 peripheral Reset condition).

Normally an Interrupt Acknowledge cycle is indicated by the Z80 CPU when /M1 and /IORQ are both active (which can be detected on the third rising clock edge after T1). To obtain an early indication of an Interrupt Acknowledge cycle, the Shift register decodes an active /M1 in the presence of an inactive /MREQ on the rising edge of T2.

During an Interrupt Acknowledge cycle, the /INTACK signal is generated on the rising edge of T2.

Since it is the presence of /INTACK and an active /READ that gates the interrupt vector onto the data bus, the logic must also generate /READ at the is Td1Ai(RD) /INTACK to /RD (Acknowledge) Low Delay]. This time delay allows the interrupt daisy chain to settle so that the device requesting the interrupt can place its interrupt vector onto the data bus. The shift register allows a sufficient time delay from the generation of /INTACK before it generates /READ. During this delay, it places the CPU into a Wait state until the valid interrupt vector can be placed onto the data bus. If the time between these two signals is insufficient for daisy chain settling, more time can be added by taking /READ and /WAIT from a later position on the Shift register.

Figure 10 illustrates Interrupt Acknowledge cycle timing resulting from the Z80A CPU to Z8500 peripheral and the Z80B CPU to A8500A peripheral interface. This timing comes from the logic illustrated in Figure 9, which can be used for both interfaces. Should more Wait states be required, the additional time can be calculated in terms of system clocks, since the CPU clock and PCLK are the same.

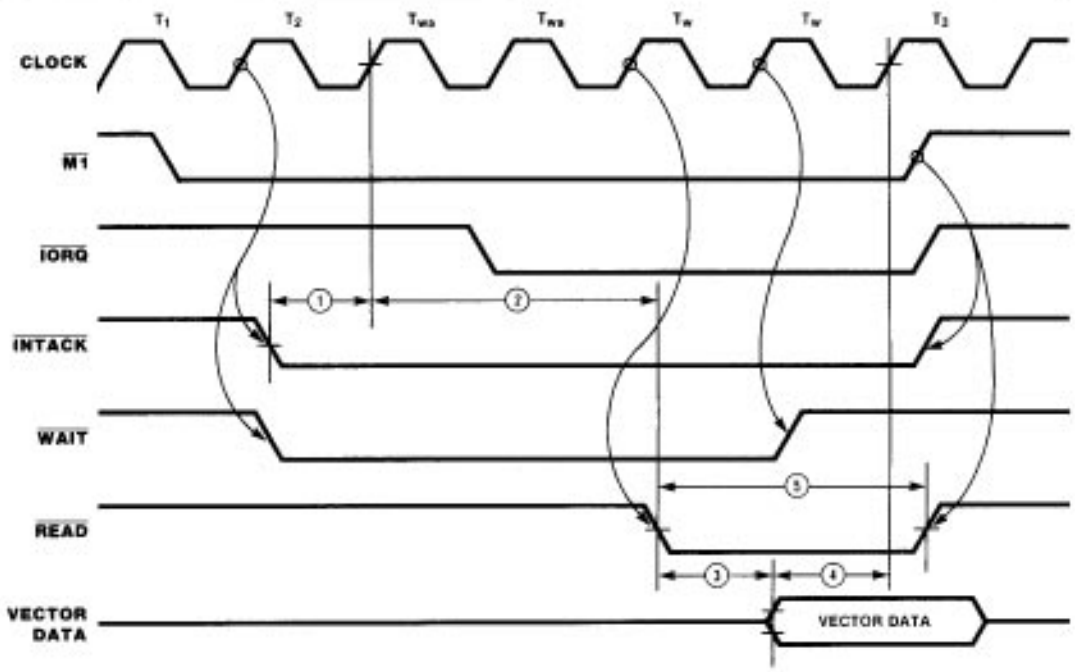



**Figure 10. Z80A/Z80B CPU to Z8500/Z8500A Peripheral Interrupt Acknowledge Interface Timing**

## Z8500/Z8500A Peripherals

Figure 11 depicts logic that can be used in interfacing the Z80H CPU to the Z8500/Z8500A peripherals. This logic is the same as that shown in Figure 5, except that a synchronizing flip-flop is used to recognize an Interrupt Acknowledge cycle. Since Z8500 peripherals do not rely upon PCLK except during Interrupt Acknowledge cycles, synchronization need occur only at that time. Since the CPU and the peripherals are running at different speeds, /INTACK and /RD must be synchronized to the Z8500 peripherals clock.

During I/O and normal memory access cycles, the synchronizing flip-flop and the Shift register remain cleared because the /M1 signal is inactive. During opcode fetch cycles, the flip-flop and the Shift register again remain cleared, but this time because the /MREQ signal is active. The synchronizing flip-flop allows an Interrupt Acknowledge cycle to be recognized on the rising edge of T2 when /M1 is active and /MREQ is inactive, generating the INTA signal. When INTA is active, the Shift register can clock and generate /INTACK to the peripheral and /WAIT to the CPU. The Shift register delays the generation of /READ to the peripheral until the daisy chain settles. The /WAIT signal is removed when sufficient time has been allowed for the interrupt vector data to be valid.

Figure 12 illustrates Interrupt Acknowledge cycle timing for the Z80H CPU to Z8500 peripheral interface. Figure 13 illustrates Interrupt Acknowledge cycle timing for the Z80H CPU to Z8500A peripheral interface. These timing result from the logic in Figure 11. Should more Wait states be required, the needed time should be calculated in terms of PCLKs, not CPU clocks.

## Z80 CPU to Z80 and Z8500 Peripherals

In a Z80 system, a combination of Z80 peripherals and Z8500 peripherals can be used compatibly. While there is no restriction on the placement of the Z8500 peripherals in the daisy chain, it is recommended that they be placed early in the chain to minimize propagation delays during RET1 cycles.

During an Interrupt Acknowledge cycle, the IEO line from Z8500 peripherals changes to reflect the interrupt status. Time should be allowed for this change to ripple through the remainder of the daisy chain before activating /IORQ to the Z80 peripherals, or /READ to the Z8500 peripherals.

## EXTERNAL INTERFACE LOGIC (Continued)

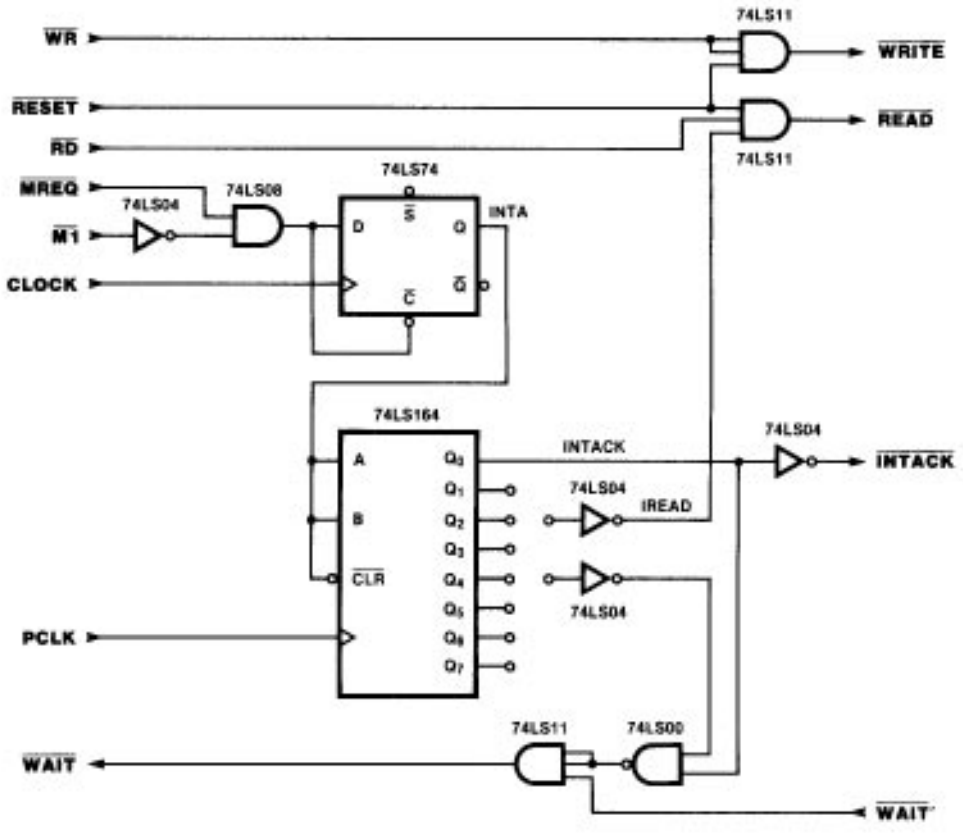



**Figure 11. Z80H to Z8500/Z8500A Peripheral Interrupt Acknowledge Interface Logic**

During RETI cycles, the IEO line from the Z8500 peripherals does not change state as in the Z80 peripherals. As long as the peripherals are at the top of the daisy chain, propagation delays are minimized.

The logic necessary to create the control signals for both Z80 and Z8500 peripherals is shown in Figure 9. This logic delays the generation of /IORQ to the Z80 peripherals by the same amount of time necessary to generate /READ for the Z8500 peripherals. Timing for this logic during an Interrupt Acknowledge cycle is depicted in Figure 10.

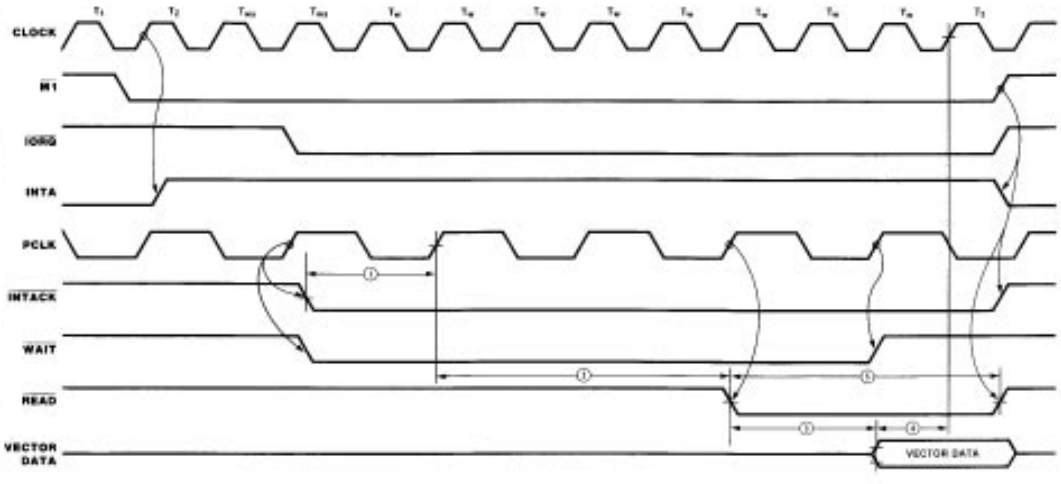



**Figure 12. Z80H CPU to Z8500 Peripheral Interrupt Acknowledge Interface Timing**

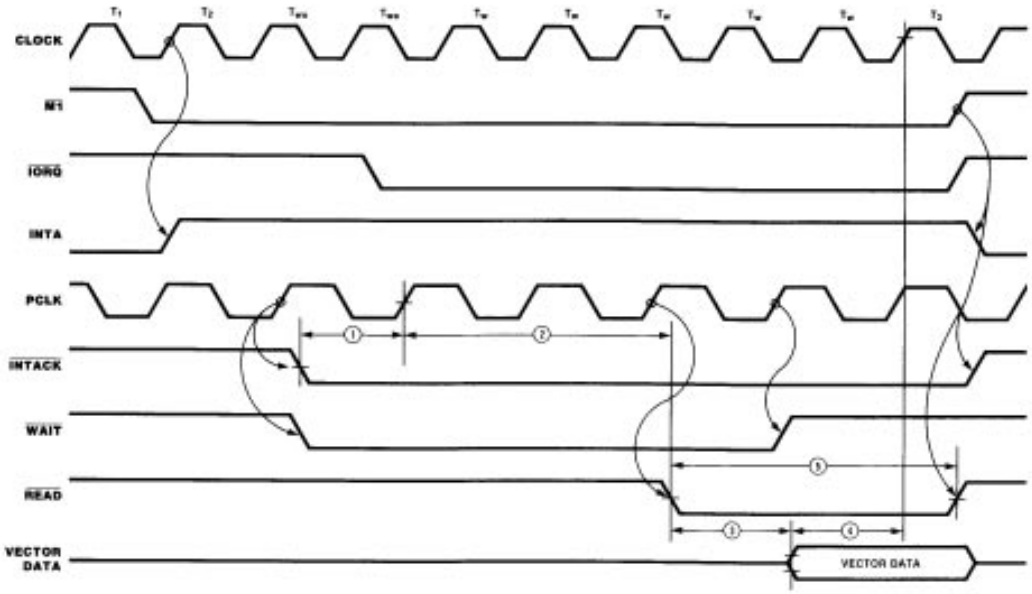



**Figure 13. Z80H CPU to Z8500A Peripheral Interrupt Acknowledge Interface Timing**

## EXTERNAL INTERFACE LOGIC (Continued)

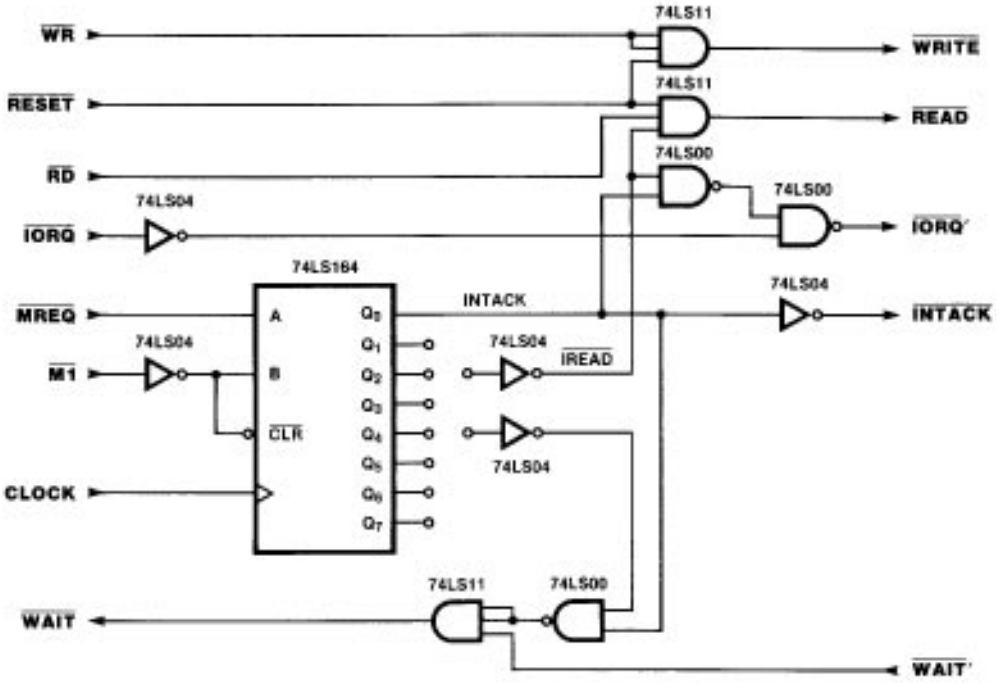



**Figure 14. Z80 and Z8500 Peripheral Interrupt Acknowledge Interface Logic**

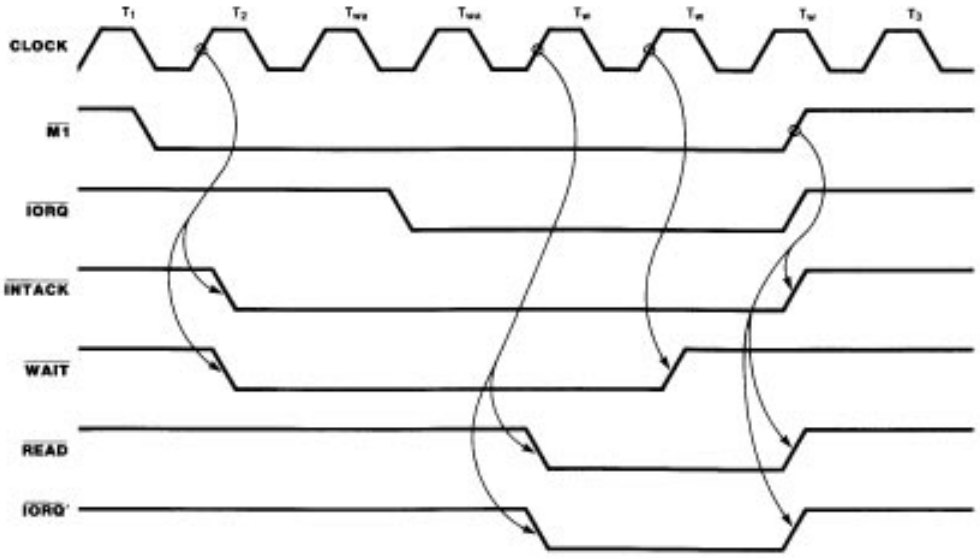



**Figure 15. Z80 and Z8500 Peripheral Interrupt Acknowledge Interface Timing**

## SOFTWARE CONSIDERATIONS - POLLED OPERATION

There are several options available for servicing interrupts on the Z8500 peripherals. Since the vector of IP registers can be read at any time, software can be used to emulate the Z80 interrupt response. The interrupt vector read reflects the interrupt status condition even if the device is programmed to return to vector that does not reflect the status change (SAV or VIS is not set). The code below is a simple software routine that emulates the Z80 vector response operation.

### Z80 Vector Interrupt Response, Emulation by Software

```
;This code emulates the Z80 vector interrupt
;operation by reading the device interrupt
;vector and forming an address from a vector
;table. It then executes an indirect jump to
;the interrupt service routine.

INDX:   LD      A,CIVREG        ;CURRENT INT. VECT. REG
        OUT     (CTRL), A       ;WRITE REG. PTR.
        IN      A, (CTRL)       ;READ VECT. REG.
        INC     A               ;VALID VECTOR?
        RET     Z               ;NO INT - RETURN
        AND     00001110B       ;MASK OTHER BITS
        LD      E,A
        LD      D,0             ;FORM INDEX VALUE
        LD      HL,VECTAB
        ADD     HL,DE           ;ADD VECT. TABLE ADDR.
        LD      A, (HL)         ;GET LOW BYTE
        INC     HL
        LD      H, (HL)         ;GET HIGH BYTE
        LD      L,A             ;FORM ROUTINE ADDR.
        JP      (HL)            ;JUMP TO IT

VECTAB:
        DEFW    INT1
        DEFW    INT2
        DEFW    INT3
        DEFW    INT4
        DEFW    INT5
        DEFW    INT6
        DEFW    INT7
        DEFW    INT8
```

## A SIMPLE Z80-Z8500 SYSTEM

The Z8500 devices interface easily to the Z80 CPU, thus providing a system of considerable flexibility. Figure 16 illustrates a simple system using the Z80A CPU and Z8536 Counter/Timer and Parallel I/O Unit (CIO) in a mode 1 or non-interrupt environment. Since interrupt vectors are not used, the /INTACK line is tied High and no additional logic is needed. Because the CIO can be used in a polled interrupt environment, the /INT pin is connected to the CPU. The Z80 should not be set for mode 2 interrupts since the CIO will never place a vector onto the data bus. Instead, the CPU should be placed into mode 1 interrupt mode and a global interrupt service routine can poll the CIO to determine what caused the interrupt to occur. In this system, the software emulation procedure described above is effective.

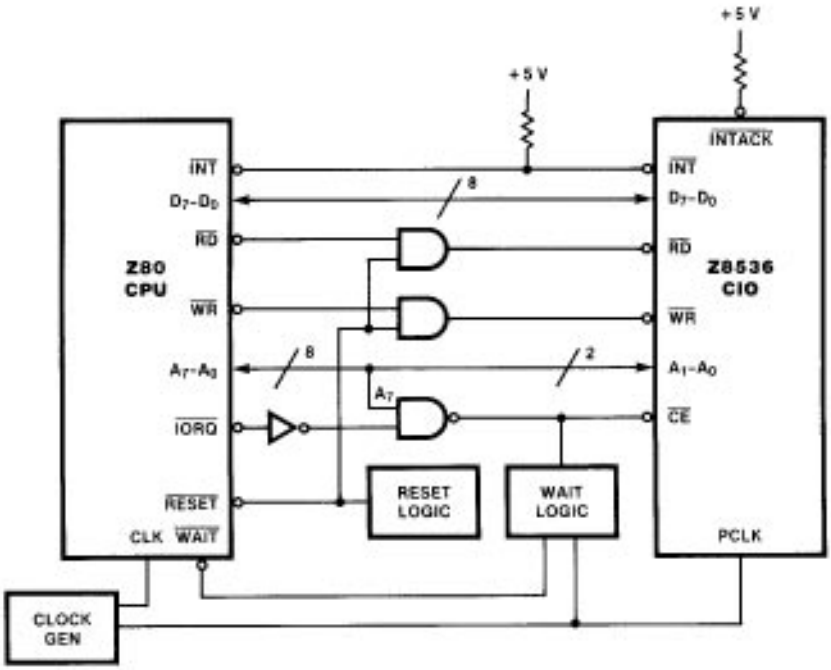



**Figure 16. Z80 to Z8500 Simple System Mode 1 Interrupt or Non-Interrupt Structure**

### Additional Information in Zilog Publications:

The Z80 Family User's Manual includes technical information on the Z80 CPU, DMA, PIO, CTC, and SIO.

Technical information on the Z80 CPU AC Characteristics and the Z80 Family Interrupt Structure Tutorial can be found in the Z80 Databook.

The Z8000 User's Manual features technical information on the Z8536 CIO and Z8038 FIO.

# THE Z180™ INTERFACED WITH THE SCC AT MHZ

*Build a simple system to prove and test the Z180 MPU interfacing the SCC at 10 MHz. Replacing the Z80 with the Z180 provides higher integration, reduced parts, more board space, increased processing speed, and greater reliability.*

## INTRODUCTION

This Application Note describes the design of a system using a Z80180 MPU (Microprocessor Unit) and a Z85C30 SCC (Serial Communications Controller), both running at 10 MHz. Hereinafter, all references are to the Z180™ and SCC.

The system board is a vehicle for demonstration and evaluation of the 10 MHz interface and includes the following parts:

- Z8018010VSC Z180 MPU 10 MHz, PLCC package
- Z85C3010VSC C-MOS Z8530 SCC Serial Communication Controller, 10 MHz, PLCC package
- 27C256 EPROM
- 55257 Static RAM

The Z180 is a Z80-compatible High Integration device with various peripherals on-board. Using this device as an alternative to the Z80 CPU, reduces the number of parts and board space while increasing processing speed and reliability.

The serial communication devices on the Z180 are: two asynchronous channels and one clocked serial channel. This means handling synchronous serial communications protocols requires an off-chip “multi-protocol serial communication controller.” The SCC is the ideal device for this purpose.

Zilog’s SCC is the multi-protocol (@ 10 MHz) universal serial communication controller which supports most serial communication applications including Monosync, Bisync and SDLC at 2.5 Mbits/sec speeds. Further, the wide acceptance of this device by the market ensures it is an “industry standard” serial communication controller. Also, the Z180 has special numbers for system clock frequencies of 6.144 - and 9.216 MHz which generate exact baud rates for on-chip asynchronous serial communication channels. This is due to the SCC’s on-chip, 16-bit wide baud rate generator for asynchronous ASCI communications.

The following 10 MHz interface explanation defines how the interrupt structure works. Also included is a discussion of the hardware and software considerations involved in running the system’s communication board. This Application Note assumes the reader has a strong working knowledge of the Z180 and SCC; this is not a tutorial for each device.

## INTERFACES

The following subsections explain the interfaces between the:

- Z180 and Memory
- Z180 and I/O
- Z180 and SCC

Basic goals of this system design are:

- System clock up to 10 MHz
- Using the Z8018010VSC (Z180 10 MHz PLCC package) to take advantage of 1M byte addressing space and compactness (DIP versions' addressing range is half; 512K bytes)
- Using Z85C3010VSC (CMOS SCC 10 MHz PLCC package)
- Minimum parts count
- Worst case design
- Using EPLD for glue wherever possible
- Expendability

The design method for EPLD is using TTLs (74HCT) and then translating them into EPLD logic. This design uses TTLs and EPLDs. With these goals in mind, the discussion begins with the Z180-to-memory interface.

### Z180 to Memory Interface

The memory access cycle timing of the Z180 is similar to the Z80 CPU memory access cycle timing. The three classifications are:

- Opcode fetch cycle (Figure 1)
- Memory read cycle (Figure 2)
- Memory write cycle (Figure 3)

Table 1 shows the Z180's basic timing elements for the opcode's fetch/memory read/write cycle.

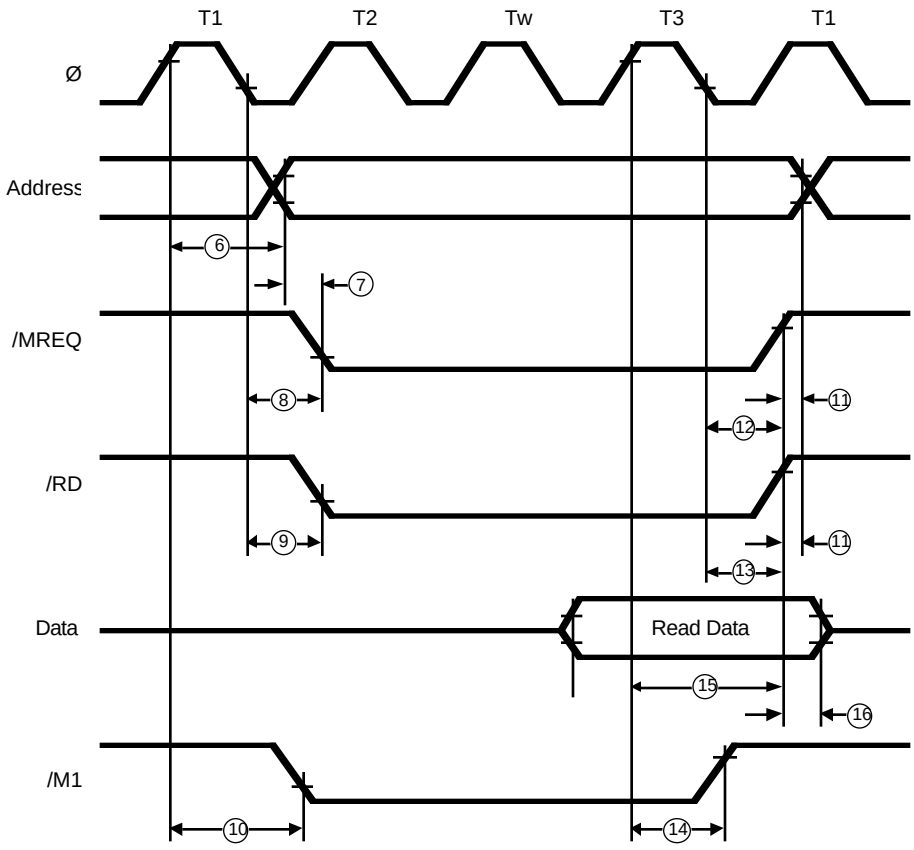



**Figure 1. Z180 Opcode Fetch Cycle Timing (One Wait State)**

**Table 1. Z8018010 Timing Parameters for Opcode Fetch Cycle (Worst Case: Z180 10 MHz)**

| No | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| 1 | tcyc | Clock Cycle Period | 100 | | ns |
| 2 | tCHW | Clock Cycle High Width | 40 | | ns |
| 3 | tCLW | Clock Cycle Low Width | 40 | | ns |
| 4 | tcf | Clock Fall Time | | 10 | ns |
| 6 | tAD | Clock High to Address Valid | | 70 | ns |
| 8 | tMED1 | Clock Low to /MREQ Low | | 50 | ns |
| 9 | tRDD1 | Clock Low to /RD Low | | 50 | ns |
| 11 | tAH | Address Hold Time | 10 | | ns |
| 12 | tMED2 | Clock Low to /MREQ High | | 50 | ns |
| 15 | tDRS | Data to Clock Setup | 25 | | ns |
| 16 | tDRH | Data Read Hold Time | 0 | | ns |
| 22 | tWRD1 | Clock High to /WR Low | | 50 | ns |
| 23 | tWDD | Clock Low to Write Data Delay | | 60 | ns |
| 24 | tWDS | Write Data Setup to /WR Low | 15 | | ns |
| 25 | tWRD2 | Clock Low to /WR High | | 50 | ns |
| 26 | tWRP | /WR Pulse Width | | 110 | ns |
| 27 | tWDH | /WR High to Data Hold Time | 10 | | ns |

**Note:** Parameter numbers in this table are in the Z180 technical manual.

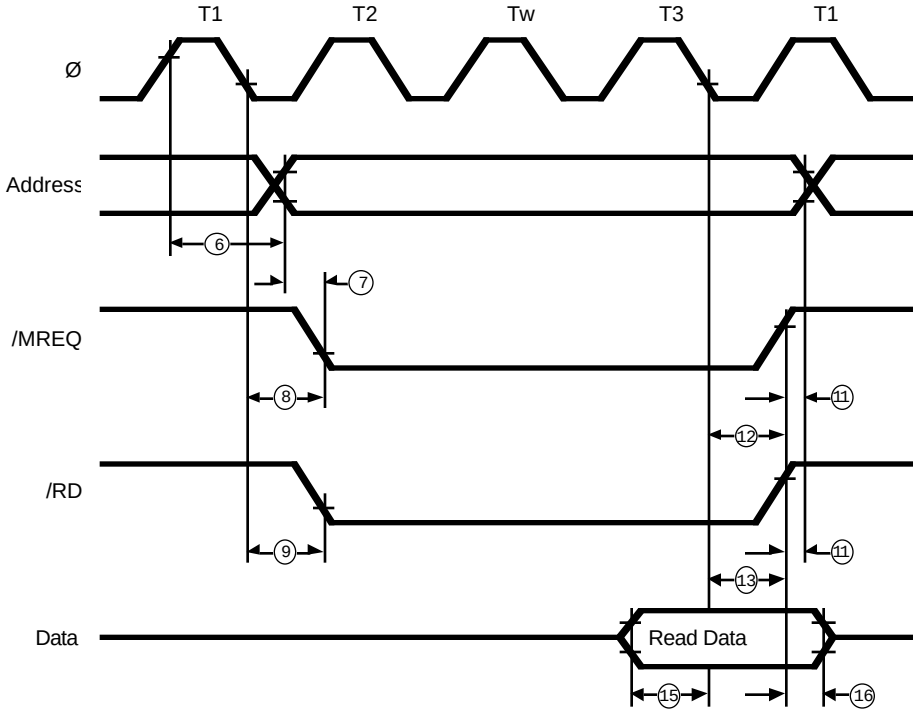



**Figure 2. Z180 Memory Read Cycle Timing (One Wait State)**

## EPROM INTERFACE

During an Opcode fetch cycle, data sampling of the bus is on the rising PHI clock edge of T3 and on the falling edge of T3 during a memory read cycle. Opcode fetch cycle data sample timing is half a clock cycle earlier. Table 2 shows how a memory read cycles' timing requirements are easier than an opcode fetch cycle by half a PHI cycle time. If the timing requirements for an Opcode fetch cycle meet specifications, the design satisfies the timing requirements for a memory read cycle.

Table 2 has some equations for an opcode fetch, memory read/write cycle.

**Table 2. Parameter Equations (10 MHz) Opcode Fetch/Memory Read/Write Cycle**

| Parameters | Z180 Equation | Value | Units |
|---|---|---|---|
| Address Valid to Data Valid (Opcode Fetch) | 2(1+w)tcyc-tAD-tDRS | 105+100w min | ns |
| Address Valid to Data Valid (Memory Read | 2(1+w)tcyc+tCHW+tcf-tAD-tDRS | 155+100w min | ns |
| /MREQ Active to Data Valid (Opcode Fetch) | (1+w)tcyc+tCLW-tMED1-tDRS | 55+100w min | ns |
| /MREQ Active to Data Valid (Memory Read) | (2+w)tcyc-tMED1-tDRS | 105+100w min | ns |
| /RD Active to Data Valid (Opcode Fetch) | (1+w)tcyc+tCLW-tRRD1-tDRS | 55+100w min | ns |
| /RD Active to Data Valid (Memory Read) | (2+w)tcyc-tRRD1-tDRS | 105+100w min | ns |
| Memory Write Cycle /WR Pulse Width | tWRP+w*tcyc | 110+100w min | ns |

**Note:** * w is the number of wait states.

The propagation delay for the decoded address and gates in the previous calculation is zero. Hence, on the real design, subtracting another 20-30 ns to pay for propagation delays, is possible. The 27C256 provides the EPROM for this board. Typical timing parameters for the 27C256 are in Table 3.

**Table 3. EPROM (27C256) Key Timing Parameters (Values May Vary Depending On Mfg.)**

| | Access Time | | |
|---|---|---|---|
| | 170 ns | 200 ns | 250 ns |
| Parameter | Max | Max | Max |
| Addr Access Time | 170 | 200 | 250 |
| /E to Data Valid | 170 | 200 | 250 |
| /OE to Data Valid | 75 | 75 | 100 |

**Note:** Table 3 shows "Access Time" as applying /E to data valid. "/OE active to data valid" is shorter than "address access time". Hence, the interface logic for the EPROM is: Realize a 170 ns or faster EPROM access time by adding one wait state (using the on-chip wait state generator of the Z180). A 200 ns requirement uses two wait states for memory access.

### SRAM Interface

Table 4 has timing parameters for 256K bit SRAM for this design.)

**Table 4. 256K SRAM Key Timing parameters (Values May Vary Depending On Mfg.)**

| | Access Time | | |
|---|---|---|---|
| | 85 ns | 100 ns | 150 ns |
| Parameter | Min | Min | Min |
| **Read Cycle:** | | | |
| /E to Data Valid | 85 | 100 | 150 |
| /G to Data Valid | 45 | 40 | 60 |
| **Write Cycle:** | | | |
| Write Cycle Time | 85 | 100 | 150 |
| Addr Valid to End of Write | 75 | 80 | 100 |
| Chip Select to End of Write | 75 | 80 | 100 |
| Data Select to End of Write | 40 | 40 | 60 |
| Write Pulse Width | 60 | 60 | 90 |
| Addr Setup Time | 0 | 0 | 0 |

**SRAM Read Cycle.** An SRAM read cycle shares the same considerations as an EPROM interface.

Like EPROM, SRAMs' "access time" applies /G to data valid, and "/E active to data valid" is shorter than "access time." This design allows the use of a 150 ns access time SRAM by adding one wait state (using the on-chip wait state generator of the Z180). The circuit is common to the EPROM memory read cycle.

No wait states are necessary if there is a 85 ns, or faster, access time by using SRAMs. Since the Z180 has on-chip MMU with 85 ns or faster SRAM just copy the contents of EPROM (application program starts at logical address 0000h) into SRAM after power on. Set up the MMU to SRAM area to override the EPROM area and stop

inserting wait states. With this scheme, you can get the highest performance with moderate cost.

**SRAM Write Cycle.** During a Z180 memory write cycle, the Z180 write data is stable before the falling edge of /WR (Z180 parameter #24; 15 ns min at 10 MHz). It is stable throughout the write cycle (Z180 parameter #27; 10 ns min at 10 MHz). Further, the address is fixed before the falling edge of /WR. As long as the /WR pulse width meets the SRAM's spec, there is no problem (reference Table 2).

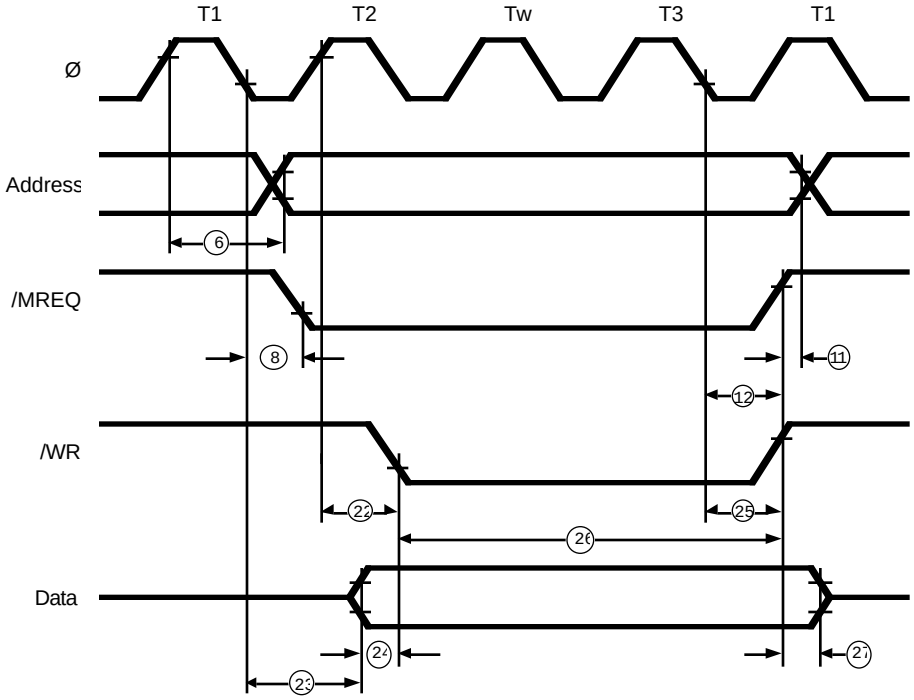



**Figure 3. Z180 Memory Write Cycle Timing (One Wait State)**

**Memory Interface Logic**

The memory devices (EPROM and SRAM) for this design are 256K bit (32K byte). There are two possible memory interface designs:

Connect Address Decode output to /E input. Put the signal generated by /RD and /MREQ ANDed together to /OE of EPROM and SRAM. Put the signal generated by /WR and /MREQ ANDed together to the /WE pin of SRAM (Figure 4a).

Connect the signal Address ANDed together with inactive /IORQ to the /E input. Connect /RD to /OE of EPROM and SRAM, and /WR to /WE pin of SRAM (Figure 4b).

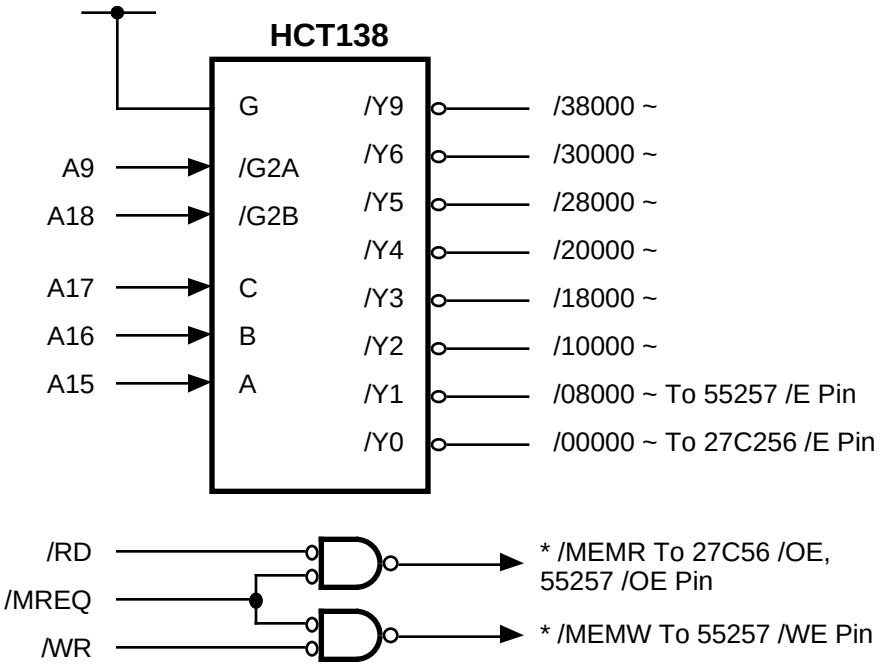
Using the second method, there could be a narrow glitch on the signal to the /E-pin during I/O cycles and the Interrupt acknowledge cycle. During I/O cycles, /IORQ and /RD or /WR go active at almost the same time. Since the delay times of these signals are similar there is no "overlapping time" between /CE generated by the address (/IORQ inactive), and /WR or /RD active. During the Interrupt Acknowledge cycle, /WR and /RD signals are inactive.

To keep the design simple and flexible, use the second method (Figure 4b). To expand memory, decode the address A15 NANDed with /USRRAM//USRROM and /IORQ to produce /CSRAM or /CSROM. These are chip select inputs to chips 55257 or 27C256, respectively. This either disables or enables on-board ROM or RAM depending upon selection control.

The circuit on Figure 4b gives the physical memory address as shown on Figure 5.

If there are no Z80 peripherals and /M1 is enabled (M1E bit in Z180 OMCR register set to 1), active wait states occur only during opcode fetch cycles (Figure 6). If the M1E bit is cleared to 0, /M1E is active only during the Interrupt Acknowledge cycle and Return from Interrupt cycle. This case depends on the propagation delay of the address decoder which uses 135 ns or faster EPROM assess time (assume there is 20 ns propagation delay). Figure 6 shows the example of this implementation.

(Continued)




* /RD to /OE Pin of 27C256 and 55257
/WR To /WE Pin of 55257

**Figure 4a. Memory Interface Logic**

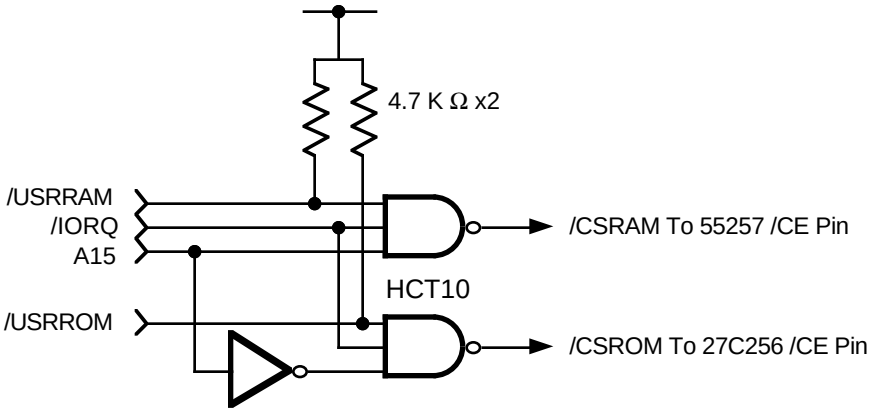



**Figure 4b. Memory Interface Logic**

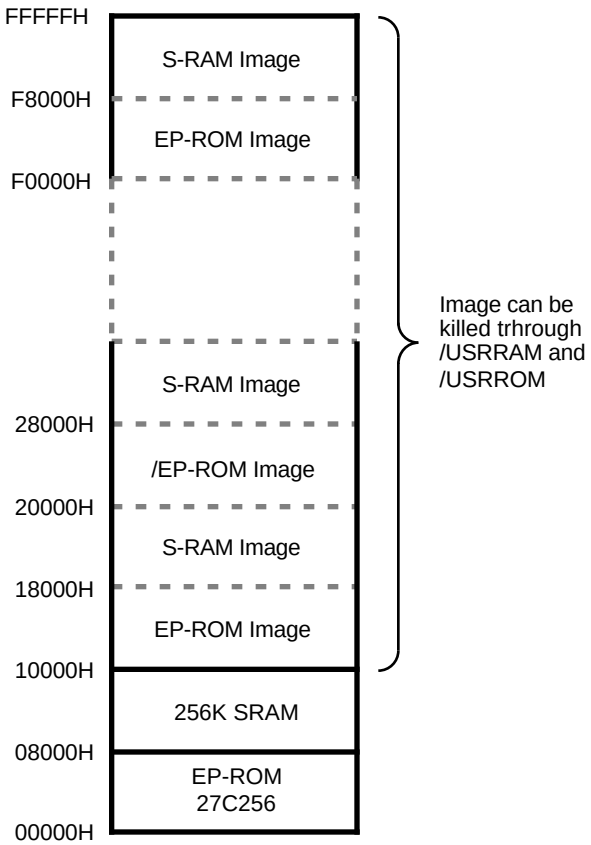



**Figure 5. Physical Memory Address Map**

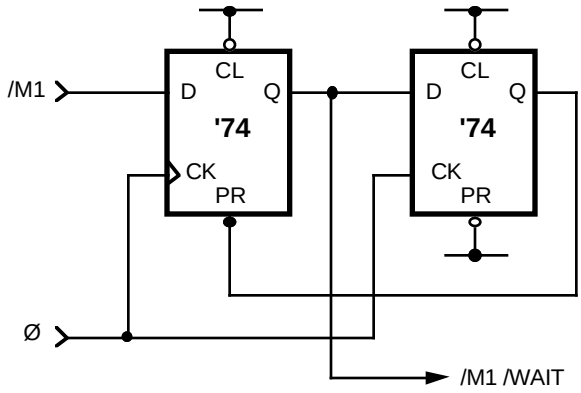



**Figure 6. Wait State Generator Logic**

(Extends Opcode Fetch Cycle Only; Not Working in Z Mode of Operation)

## Z180 TO I/O INTERFACE

The Z180 I/O read/write cycle is similar to the Z80 CPU if you clear the /IOC bit in the OMCR register to 0 (Figures 7 and 8). Table 5 shows the Z180 key parameters for an I/O cycle.

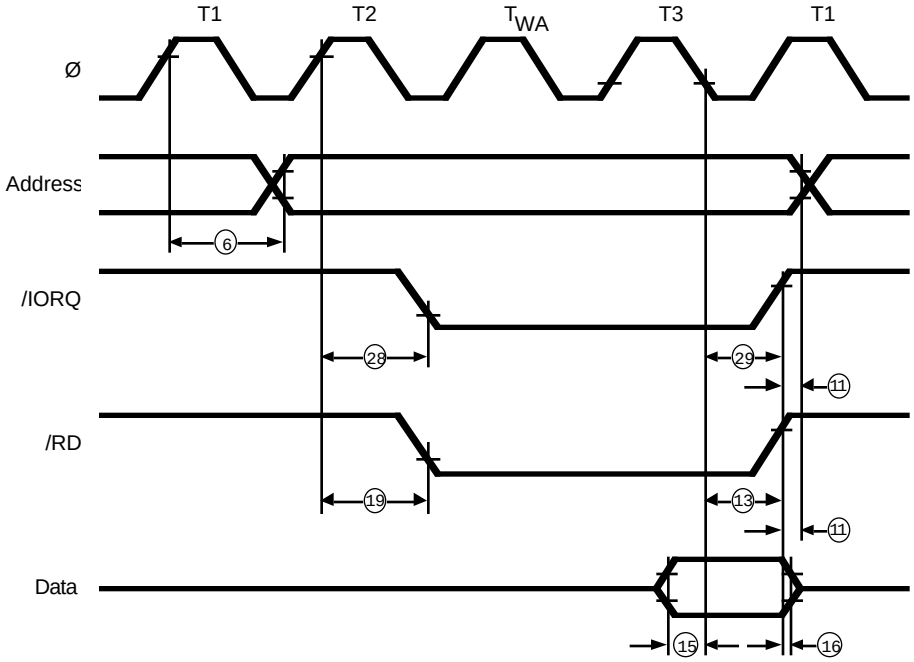



**Figure 7. Z180 I/O Read Cycle Timing (/IOC = 0)**

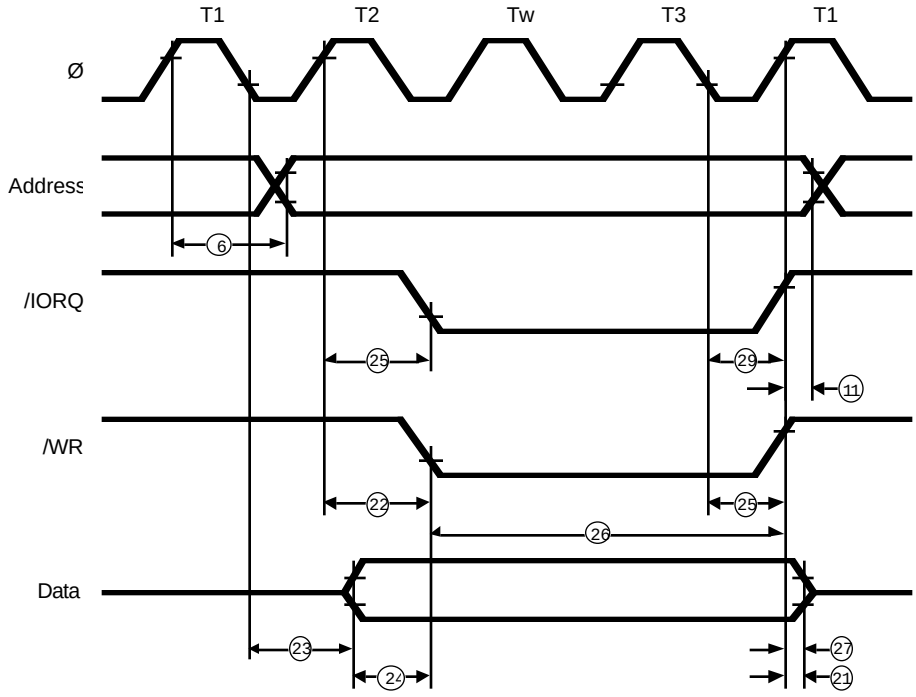



**Figure 8. Z180 I/O Write Cycle Timing**

.

**Table 5. Z8018010 Timing Parameters for I/O Cycle (Worst Case)**

| No | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| **1** | tcyc | Clock Cycle Period | 100 | | ns |
| **2** | tCHW | Clock Cycle High Width | 40 | | ns |
| **3** | tCLW | Clock Cycle Low Width | 40 | | ns |
| **4** | tcf | Clock Fall Time | | 10 | ns |
| **6** | tAD | Clock High to Address Valid | | 70 | ns |
| **9** | tRDD1 | Clock High to /RD Low IOC=0 | | 55 | ns |
| **11** | tAH | Address Hold Time | 10 | | ns |
| **13** | tRDD2 | Clock Low to /RD High | | 50 | ns |
| **15** | tDRS | Data to Clock Setup | 25 | | ns |
| **16** | tDRH | Data Read Hold Time | 0 | | ns |
| **21** | tWDZ | Clock High to Data Float Delay | | 60 | ns |
| **22** | tWRD1 | Clock High to /WR Low | | 50 | ns |
| **23** | tWDD | Clock Low to Write Data Delay | | 60 | ns |
| **24** | tWDS | Write Data Setup to /WR Low | 15 | | ns |
| **25** | tWRD2 | Clock Low to /WR High | | 50 | ns |
| **26a** | tWRP | /WR Pulse Width (I/O Write) | 210 | | ns |
| **27** | tWDH | /WR High to Data Hold Time | 10 | | ns |
| **28** | tIOD1 | Clock High to /IORQ Low IOC=0 | | 55 | ns |
| **29** | tIOD2 | Clock Low to /IORQ High | | 50 | ns |

**Note:** Parameter numbers in this table are the numbers in the Z180 technical manual.

If you are familiar with the Z80 CPU design, the same interfacing logic applies to the Z180 and I/O interface (see Figure 9a). This circuit generates /IORD (Read) or IORD (Write) for peripherals from inputs /IORQ, /RD, and /WR. The address decodes the Chip Select signal. Note, if you have Z80 peripherals, the decoder logic decodes only from addresses (does not have /IORQ). The Z180 signals /IORQ, /RD, and /WR are active at about the same time (Parameters #9, 22, 28). However, most of the Z80 peripherals require /CE to /RD or /WR setup time.

Since the Z180 occupies 64 bytes of I/O addressing space for system control and on-chip peripherals, there are no overlapping I/O addresses for off-chip peripherals. In this design, leave the area as default or assign on-chip registers at I/O address 0000h to 003Fh.

Figure 9 shows a simple address decoder (the required interface signals, other than address decode outputs, are discussed later).

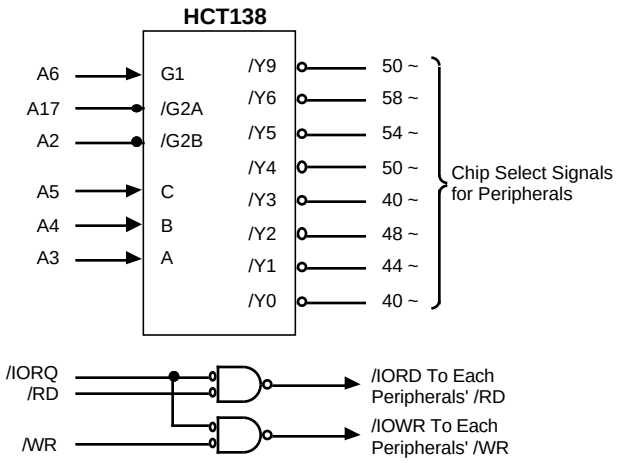



**Figure 9a. I/O Interface Logic (Example)**

(Continued)

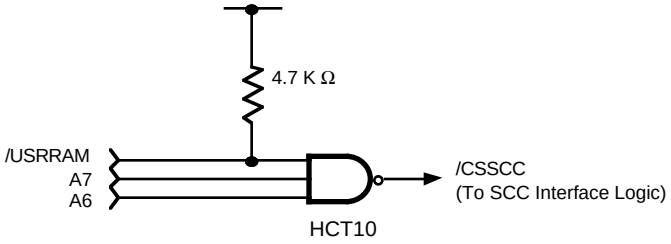



**Figure 9b. I/O Address Decoder for this Board**

When expanding this board to enable other peripherals, the decoded address A6/A7 is NANDed with USRIO to produce the Chip Enable (CSSCC) output signal (HC10). The SCC registers are assigned from address xxC0h to xxC3h; with image, they occupy xxC0h to xxFFh. To add wait states during I/O transactions, use the Z180 on-chip wait state generator instead of external hardware logic.

If there is a Z80 PIO on board in a Z-mode of operation (that is, clear /M1E in OMCR register to zero) and after enabling a Z80 PIO interrupt, zero is written to M1TE in the OMCR register. Without a zero, there is no interrupt from the Z80 PIO. The Z80 PIO requires /M1 to activate an interrupt circuit after enabling interrupt by software.

## Z180 TO SCC INTERFACE

The following subsections discuss the various parameters between the Z180/SCC interface: CPU hardware, I/O operation (read/write), SCC interrupts, Z80 interrupt daisy-chain operation, SCC interrupt daisy-chain operation, I/O cycles.

### CPU Hardware Interfacing

The hardware interface has three basic groups of signals: Data bus, system control, and interrupt control. For more detailed signal information, refer to Zilog's Technical Manuals, and Product Specifications for each device.

#### Data Bus Signals

**D7-D0.** *Data bus* (Bidirectional, tri-state). This bus transfers data between the Z180 and SCC.

#### System Control Signals

**A//B, C//D.** *Register select signals* (Input). These lines select the registers.

**/CE.** *Chip enable* (Input, active low). /CE selects the proper peripheral for programming. /CE is gated with /IORQ or /MREQ to prevent false chip selects during other machine cycles.

**/RD+.** *Read* (input, active low). /RD activates the chip-read circuitry and gates data from the chip onto the data bus.

**/WR+.** *Write* (Input, active low). /WR strobes data from the data bus into the peripheral.

Chip reset occurs when /RD and /WR are active simultaneously.

#### Interrupt Control

**/INTACK.** *Interrupt Acknowledge* (input, active low). This signal shows an Interrupt Acknowledge cycle which combines with /RD to gate the interrupt vector onto the data bus.

**/INT.** *Interrupt request* (output, open-drain, active low).

**IEI.** *Interrupt Enable In* (input, active high).

**IEO.** *Interrupt Enable Out* (Output, active high).

These lines control the interrupt daisy chain for the peripheral interrupt response.

### SCC I/O Operation

The SCC generates internal control signals from /RD or /WR. Since PCLK has no required phase relationship to /RD or /WR, the circuitry generating these signals provides time for meta stable conditions to disappear.

The SCC starts the different operating modes by programming the internal registers. Accessing these internal registers occurs during I/O Read and Write cycles, described below.

#### Read Cycle Timing

Figure 10 illustrates the SCC Read cycle timing. All register addresses and /INTACK are stable throughout the cycle. The timing specification of SCC requires that the /CE signal (and address) be stable when /RD is active.

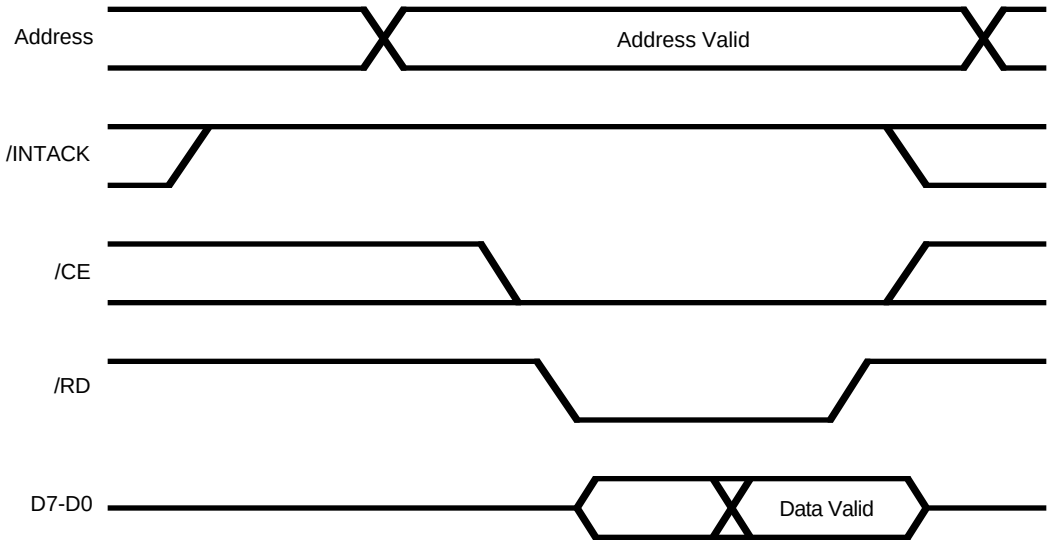



**Figure 10. SCC Read Cycle Timing**

**Write Cycle Timing**

Figure 11 illustrates the SCC Write cycle timing. All register addresses and /INTACK are stable throughout the cycle. The timing specification of the SCC requires that the /CE signal (and address) be stable when /RD is active. Data is available to the SCC before the falling edge of /WR and remains active until /WR goes inactive.

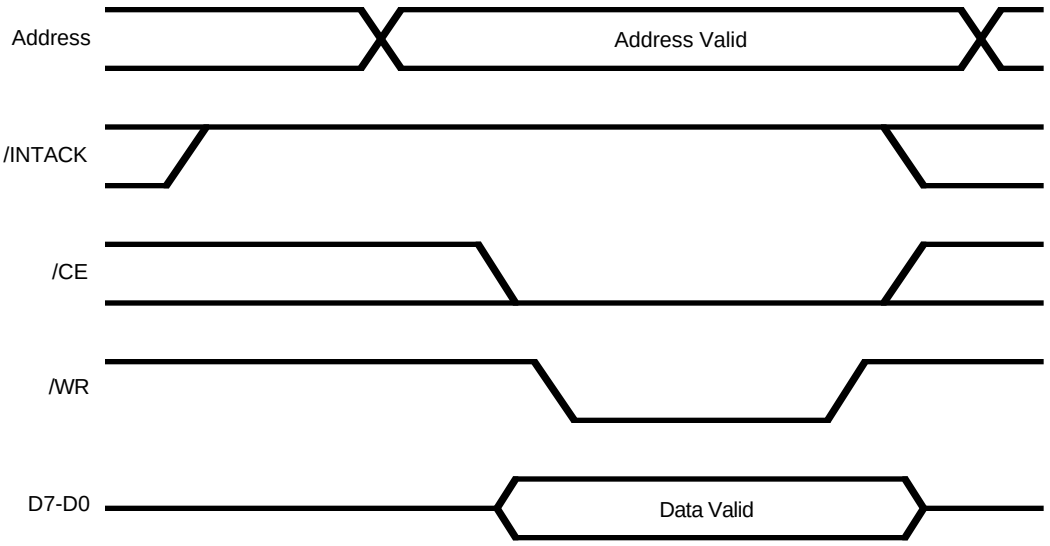



**Figure 11. SCC Write Cycle Timing**

(Continued)

## SCC Interrupt Operation

Understanding SCC interrupt operations requires a basic knowledge of the Interrupt Pending (IP) and Interrupt Under Service (IUS) bits in relation to the daisy chain. The Z180 and SCC design allow no additional interrupt requests during an Interrupt Acknowledge cycle. This permits the interrupt daisy chain to settle, ensuring proper response of the interrupt device.

The IP bit sets in the SCC for CPU intervention requirements (that is, buffer empty, character available, error detection, or status changes). The interrupt acknowledge cycle does not reset the IP bit. The IP bit clears by a software command to the SCC, or when the action that generated the interrupt ends, for example, reading a receive character for receive interrupt. Others are, writing data to the transmitter data register, issuing Reset Tx interrupt pending command for Tx buffer empty interrupt, etc.). After servicing the interrupt, other interrupts can occur.

The IUS bit means the CPU is servicing an interrupt. The IUS bit sets during an Interrupt Acknowledge cycle if the IP bit sets and the IEI line is High. If the IEI line is low, the IUS bit is not set. This keeps the device from placing its vector onto the data bus.

The IUS bit clears in the Z80 peripherals by decoding the RETI instruction. A software command also clears the IUS bit in the Z80 peripherals. Only software commands clear the IUS bit in the SCC.

## Z80 Interrupt Daisy-Chain Operation

In the Z80 peripherals, both IP and IUS bits control the IEO line and the lower portion of the daisy chain. When a peripheral's IP bit sets, the IEO line goes low. This is true regardless of the state of the IEI line. Additionally, if the peripheral's IUS bit clears and its IEI line is High, the /INT line goes low.

The Z80 peripherals sample for both /M1 and /IORQ active (and /RD inactive) to identify an Interrupt Acknowledge cycle. When /M1 goes active and /RD is inactive, the peripheral detects an Interrupt Acknowledge cycle and allows its interrupt daisy chain to settle. When the /IORQ line goes active with /M1 active, the highest priority interrupting peripheral places its interrupt vector onto the data bus. The IUS bit also sets to show that the peripheral is now under service. As long as the IUS bit sets, the IEO line remains low. This inhibits any lower priority devices from requesting an interrupt.

When the Z180 CPU executes the RETI instruction, the peripherals check the data bus and the highest priority device under service resets its IUS bit.

## SCC Interrupt Daisy-Chain Operation

In the SCC, the IUS bit normally controls the state of the IEO line. The IP bit affects the daisy chain only during an Interrupt Acknowledge cycle. Since the IP bit is normally not part of the SCC interrupt daisy chain, there is no need to decode the RETI instruction. To allow for control over the daisy chain, the SCC has a Disable Lower Chain (DLC) software command that pulls IEO low. This selectively deactivates parts of the daisy chain regardless of the interrupt status. Table 6 shows the truth table for the SCC interrupt daisy chain control signals during certain cycles. Table 12 shows the interrupt state diagram for the SCC.

**Table 6. SCC Daisy Chain Signal Truth Table**

| During Idle State | | | | During INTACK Cycle | | | |
|---|---|---|---|---|---|---|---|
| **IEI** | **IP** | **IUS** | **IEO** | **IEI** | **IP** | **IUS** | **IEO** |
| 0 | X | X | 0 | 0 | X | X | 0 |
| 1 | X | 0 | 1 | 1 | 1 | X | 0 |
| 1 | X | 1 | 0 | 1 | X | 1 | 0 |
| 1 | 0 | 0 | 1 | | | | |

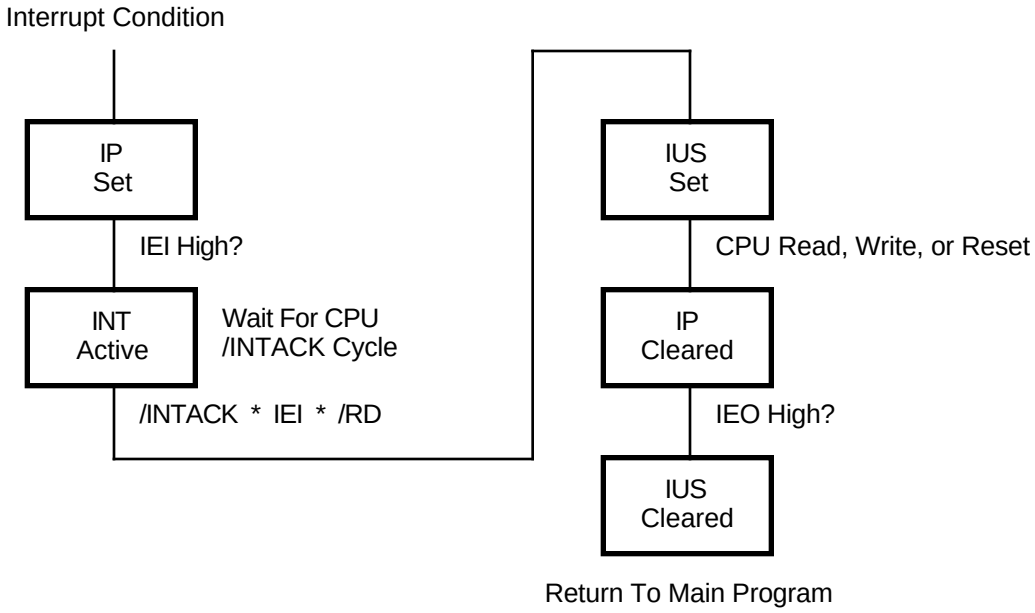



**Figure 12. SCC Interrupt Status Diagram**

The SCC uses /INTACK (Interrupt Acknowledge) for recognition of an interrupt acknowledge cycle. This pin, used with /RD, allows the SCC to gate its interrupt vector onto the data bus. An active /RD signal during an interrupt acknowledge cycle performs two functions. First, it allows the highest priority device requesting an interrupt to place its vector on the data bus. Secondly, it sets the IUS bit in the highest priority device to show the device is now under service.

## INPUT/OUTPUT CYCLES

Although the SCC is a universal design, certain timing parameters differ from the Z180 timing. The following subsections discuss the I/O interface for the Z180 MPU and SCC.

### Z180 MPU to SCC Interface

Table 7 shows key parameters of the 10 MHz SCC for I/O read/write cycles.

**Table 7. 10 MHz SCC Timing Parameters for I/O Read/Write Cycle (Worst Case)**

| No | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| 6 | TsA(WR) | Address to /WR Low Setup | 50 | | ns |
| 7 | ThA(WR) | Address to /WR High Hold | 0 | | ns |
| 8 | TsA(RD) | Address to /RD Low Setup | 50 | | ns |
| 9 | ThA(RD) | Address to /RD High Hold | 0 | | ns |
| 16 | TsCEI(WR) | /CE Low to /WR Low Setup | 0 | | ns |
| 17 | ThCE(WR) | /CE to /WR High Hold | 0 | | ns |
| 19 | TsCEI(RD) | /CE Low to /RD Low Setup | 0 | | ns |
| 20 | ThCE(RD) | /CE to /RD High Hold | 0 | | ns |
| 22 | TwRDI | /RD Low Width | 125 | | ns |
| 25 | TdRDf(DR) | /RD Low to Read Data Valid | | 120 | ns |
| 27 | TdA(DR) | Address to Read Data Valid | | 180 | ns |
| 28 | TwWRI | /WR Low Width | 125 | | ns |
| 29 | TsDW(WR) | Write Data to /WR Low Setup | 10 | | ns |
| 30 | TdWR(W) | Write Data to /WR High Hold | 0 | | ns |

### SCC I/O Read/Write Cycle

Assume that the Z180 MPU's /IOC bit in the OMCR (Operation Mode Control Register) clears to 0 (this condition is a Z80 compatible timing mode for /IORQ and /RD). The following are several design points to consider (also see Table 3).

**I/O Read Cycle**

Parameters 8 and 9 mean that Address is stable 20 ns before the falling edge of /RD and until /RD goes inactive.

Parameters 19 and 20 mean that /CE is stable at the falling edge of /RD and until /RD goes inactive.

Parameter 22 means the /RD pulse width is wider than 125 ns.

Parameters 25 and 27 mean that Read data is available on the data bus 120 ns later than the falling edge of /RD and 180 ns from a stable Address.

**I/O Write Cycle**

Parameters 6 and 7 mean that Address is stable 50 ns before the falling edge of /WR and is stable until /WR goes inactive.

Parameters 16 and 17 mean that /CE is stable at the falling edge of /WR and is stable until /W goes inactive.

Parameter 28 means /WR pulse width is wider than 125 ns.

Parameters 28 and 29 mean that Write data is on the data bus 10 ns before the falling edge of /WR. It is stable until the rising edge of /WR.

Tables 8 and 9 show the worst case SCC parameters calculating Z180 parameters at 10 MHz.

**Table 8. Parameter Equations Worst Case (Without Delay Signals - No Wait State)**

| SCC Parameters | Z180 Equation | Value | Units |
|---|---|---|---|
| TsA(RD) | tcyc-tAD+tRDD1 | 30 min | ns |
| TdA(DR) | 3tcyc+tCHW+tcf-tAD-tDRS | 245 min | ns |
| TdRDf(DR) | 2tcyc+tCHW+tcf-tRDD1-tDRS | 160 min | ns |
| TwRDI | 2tcyc+tCHW+tcf-tDRS+tRDD2 | 185 min | ns |
| TsA(WR) | tcyc-tAD+tWRD1 | 30 min | ns |
| TsDW(WR) | tWDS | 15 min | ns |
| TwWRI | tWRP | 210 min | ns |

**Table 9. Parameter Equations**

| Z180 Parameters | SCC Equation | Value | Units |
|---|---|---|---|
| tDRS | Address | | |
| | 3tcyc+tCHW-tAD-TdA(DR) | 241 min | ns |
| | RD | | |
| | 2tcyc+tCHW-tRDD1-TdRD(DR) | 184 min | ns |

**I/O Read Cycle**

These tables show that a delay of the falling edge of /RD satisfies the SCC TsA(RD) timing requirement of 50 ns min. The Z180 calculated value is 30 ns min for the worst case. Also, Z180 timing specification tAH (Address Hold time) is 10 ns min. The SCC timing parameters ThA(RD) {Address to /RD High Hold} and ThCE(RD) {/CE to /RD High Hold} are minimum at 0 ns. The rising edge of /RD is early to guarantee these parameters when considering address decoders and gate propagation delays.

**I/O Write Cycle**

Delay the falling edge of /WR to satisfy the SCC TsA(/WR) timing requirement of 50 ns min. The Z180 calculates 30 ns min worst case. Further, the Z180 timing specifications tAH (Address Hold time) and tWDH (/WR high to data hold time) are both 10 ns min. The SCC timing parameters ThA(WR) {Address to /WR High Hold}, ThCE(WR) {/CE to /WR High Hold} and TdWR(W) {Write data to /WR High hold} are a minimum of 0 ns. The rising edge of /WR is early to guarantee these parameter requirements.

This circuit depicts logic for the I/O interface and the Interrupt Acknowledge Interface for 10 MHz clock of operation. Figure 13 is the I/O read/write timing chart (discussions of timing considerations on the Interrupt Acknowledge cycle and the circuit using EPLD occur later).

(Continued)

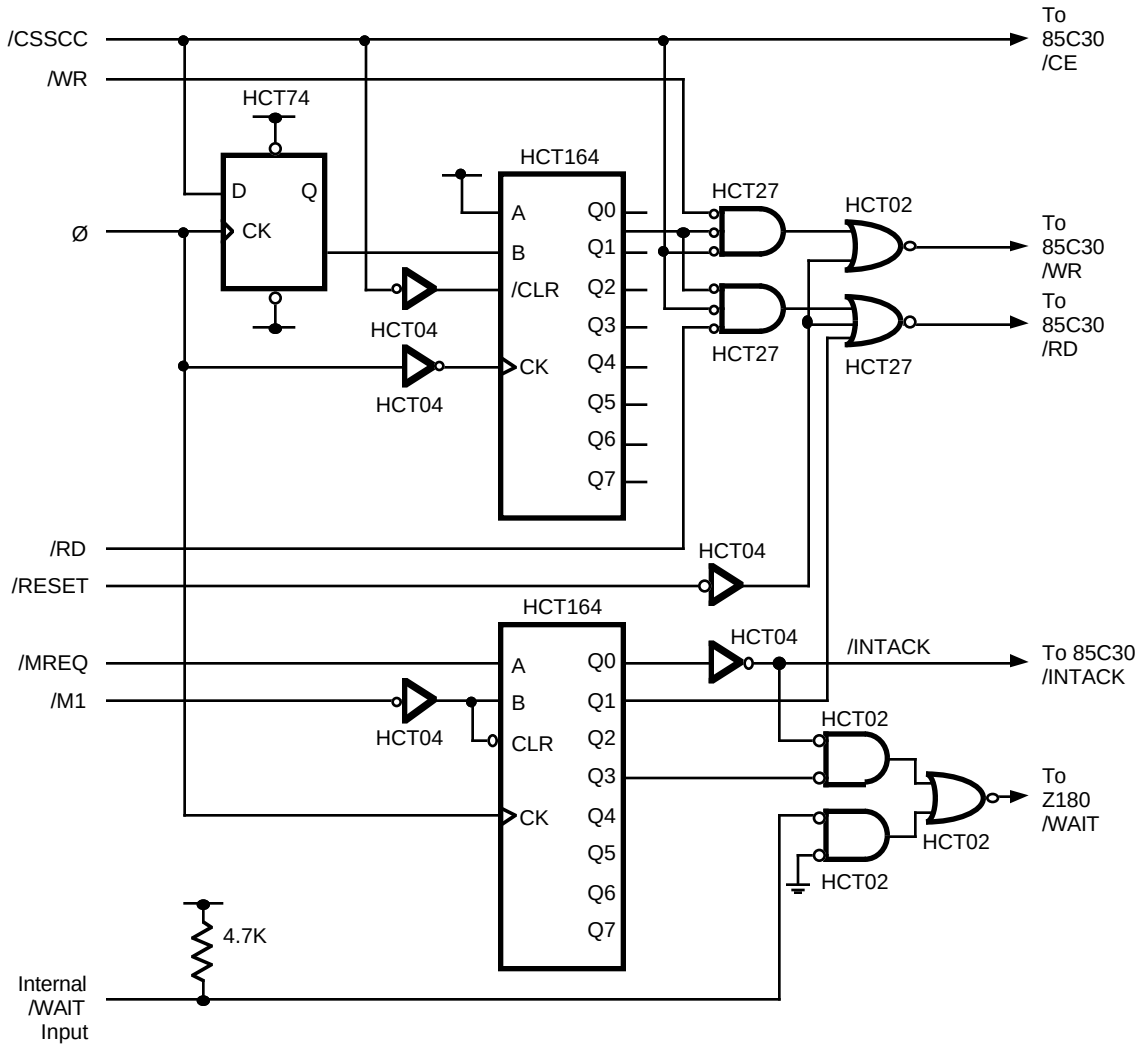



**Figure 13. SCC I/O Read/Write Cycle Timing**
This circuit works when [(Lower HCT164's CLK ≠ to Z180 /WAIT≠) + tws <tCHW]

If you are running your system slower than 8 MHz, remove the HCT74, D-Flip/Flop in front of HCT164. Connect the inverted CSSCC to the HCT164 B input. This is a required Flip/Flop because the Z180 timing specification on tIOD1 (Clock High to /IORQ Low, IOC=0) is maximum at 55 ns This is longer than half the PHI clock cycle. Sample it using the rising edge of clock, otherwise, HCT164 does not generate the same signals.

The RESET signal feeds the SCC /RD and /WR through HCT27 and HCT02 to supply the hardware reset signal. To reduce the gate count, drop these gates and make the SCC reset by its software command. The SCC software reset - 0C0h to Write Register 9, "Hardware Reset command" has the same effect as hardware reset by "Hardware."

## Interrupt Acknowledge Cycle Timing

The primary timing differences between the Z180 and SCC occur in the Interrupt Acknowledge cycle. The SCC timing parameters that are significant during Interrupt Acknowledge cycles are in Table 10. The Z180 timing parameters are in Table 10. The reference numbers in Tables 10 and 11 refer to Figure 13.

**Table 10. 10 MHz SCC Timing Parameters for Interrupt Acknowledge Cycle**

| No | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| 13 | TsIAi(RD) | /INTACK Low to /RD Low Setup | 130 | | ns |
| 14 | ThIA(RD) | /INTACK High to /RD High Hold | 0 | | ns |
| 15 | ThIA(PC) | /INTACK to PCLK High Hold | 30 | | ns |
| 38 | TwRDA | /INTACK Low to /RD Low Delay (Acknowledge) | 125 | | ns |
| 39 | TwRDA | /RD (Acknowledge) Width | 125 | | ns |
| 40 | TdRDA(DR) | /RD Low (Acknowledge) to Read Data Valid Delay | | 120 | ns |
| 41 | TsIEI(RDA) | IEI to /RD Low (Acknowledge) Setup Time | 95 | | ns |
| 42 | ThIEI(RDA) | IEI to /RD High (Acknowledge) Hold Time | 0 | | ns |
| 43 | TdIEI(IEO) | IEI to IEO Delay | | 175 | ns |

**Table 11. Z180 Timing Parameters Interrupt Acknowledge Cycles (Worst Case Z180)**

| No | Symbol | Parameter | Min | Max | Units |
|---|---|---|---|---|---|
| 10 | tM1D1 | Clock High to /M1 Low | | 60 | ns |
| 14 | tM1D2 | Clock High to /M1 High | | 60 | ns |
| 15 | tDRS | Data to Clock Setup | 25 | | ns |
| 16 | tDRH | Data Read Hold Time | 0 | | ns |
| 28 | tIOD1 | Clock LOW to /IORQ Low | | 50 | ns |
| 29 | tIOD2 | Clock LOW to /IORQ High | | 50 | ns |
| 30 | tIOD3 | /M1 Low to /IORQ Low Delay | 200 | | ns |

**Note:** Parameter numbers in this table are the numbers in the Z180 technical manual.

During an Interrupt Acknowledge cycle, the SCC requires both /INTACK and /RD to be active at certain times. Since the Z180 does not issue either /INTACK or /RD, external logic generates these signals.

The Z180 is in a Wait condition until the vector is valid. If there are other peripherals added to the interrupt priority daisy chain, more Wait states may be necessary to give it time to settle. Allow enough time between /INTACK active and /RD active for the entire daisy chain to settle.

There is no need of decoding the RETI instruction used by the Z80 peripherals since the SCC daisy chain does not use IP, except during Interrupt Acknowledge. The SCC and other Z8500 peripherals have commands that reset the individual IUS flag.

External Interface for Interrupt Acknowledge Cycle: The bottom half of Figure 14 is the interface logic for the Interrupt Acknowledge cycle.

(Continued)

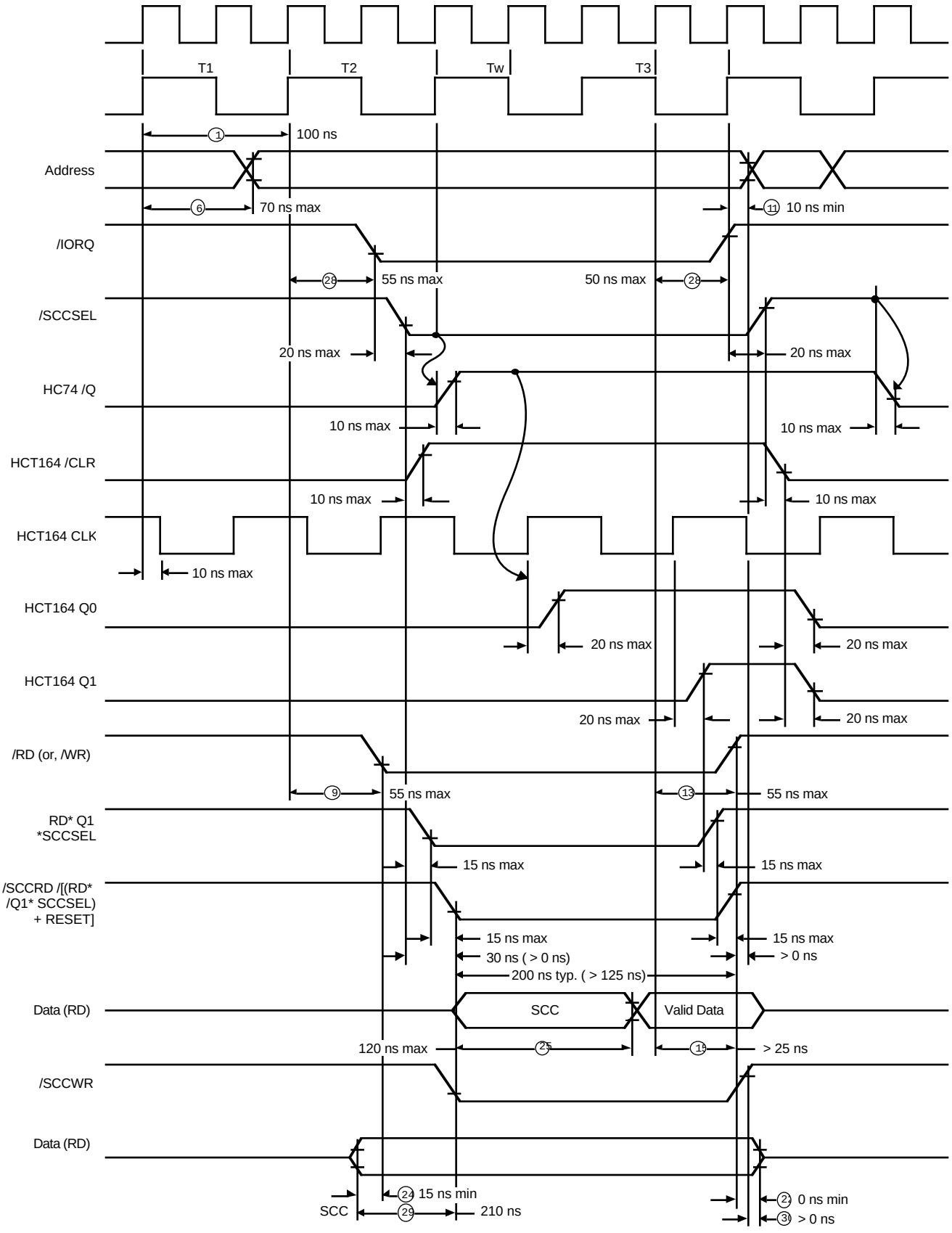



**Figure 14. Z180 to SCC Interface Logic (Example)**

The primary chip in this logic is the Shift register (HCT164), which generates /INTACK, /SCCRD and /WAIT. During I/O and normal memory access cycles, the Shift Register (HCT164) remains cleared because the /M1 signal is inactive during the opcode fetch cycle. Since the Shift Register output is Low, control of /SCCRD and /WAIT is by other system logic and gated through the NOR gate (HCT27). During I/O and normal memory access cycles, /SCCRD and /SCCWR are generated from the system /RD and /WR signals, respectively. The generation is by the logic at the top of Figure 15.

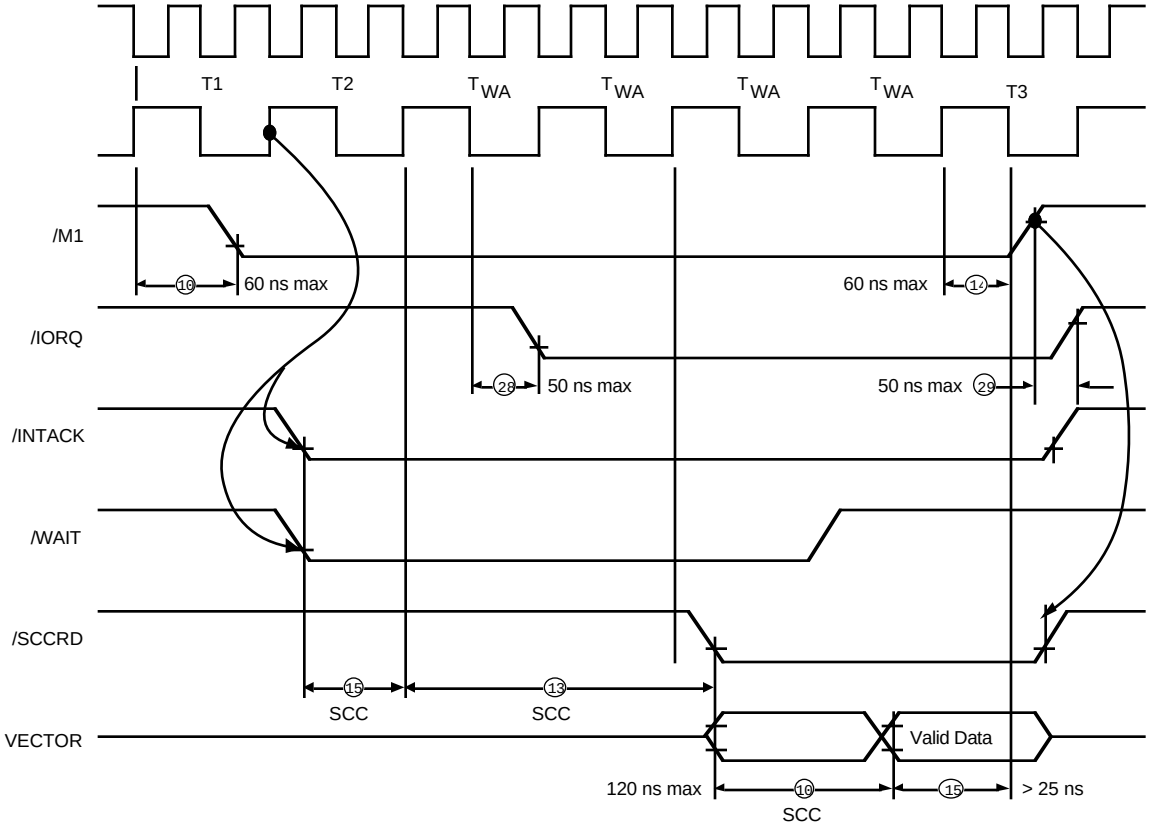



**Figure 15. SCC Interrupt Acknowledge Cycle Timing**

Normally, an Interrupt Acknowledge cycle appears from the Z180 during /M1 and /IORQ active (which is detected on the third rising edge of PHI after T1). To get an early sign of an Interrupt Acknowledge cycle, the Shift register decodes an active /M1. This is during the presence of an inactive /MREQ on the rising edge of T2.

During an Interrupt Acknowledge cycle, the /INTACK signal is generated on the rising edge of T2. Since it is the presence of /INTACK and an active SCCRD that gates the interrupt vector onto the data bus, the logic also generates /SCCRD at the proper time. The timing parameter of concern here is TdIAi(RD) [/INTACK to /RD (Acknowledge) Low delay]. This time delay allows the interrupt daisy chain to settle so the device requesting the interrupt places its interrupt vector onto the data bus.

The Shift Register allows enough time delay from the generation of /INTACK before it generates /SCCRD. During this delay, it places the Z180 into a Wait state until the valid interrupt vector is placed onto the data bus. If the time between these two signals is not enough for daisy chain settling, more time is added by taking /SCCRD and /WAIT from a later position on the Shift Register. If there is a requirement for more wait states, the time is calculated by PHI cycles.

## USING EPLD

Figure 16a and Figure 16b show the logic using either EPLD or the circuit of this system. The EPLD is ALTERA 610 which is a 24-Pin EPLD. The method to convert random gate logic to EPLD is to disassemble MSIs' logic into SSI level, and then simplify the logic.

(Continued)

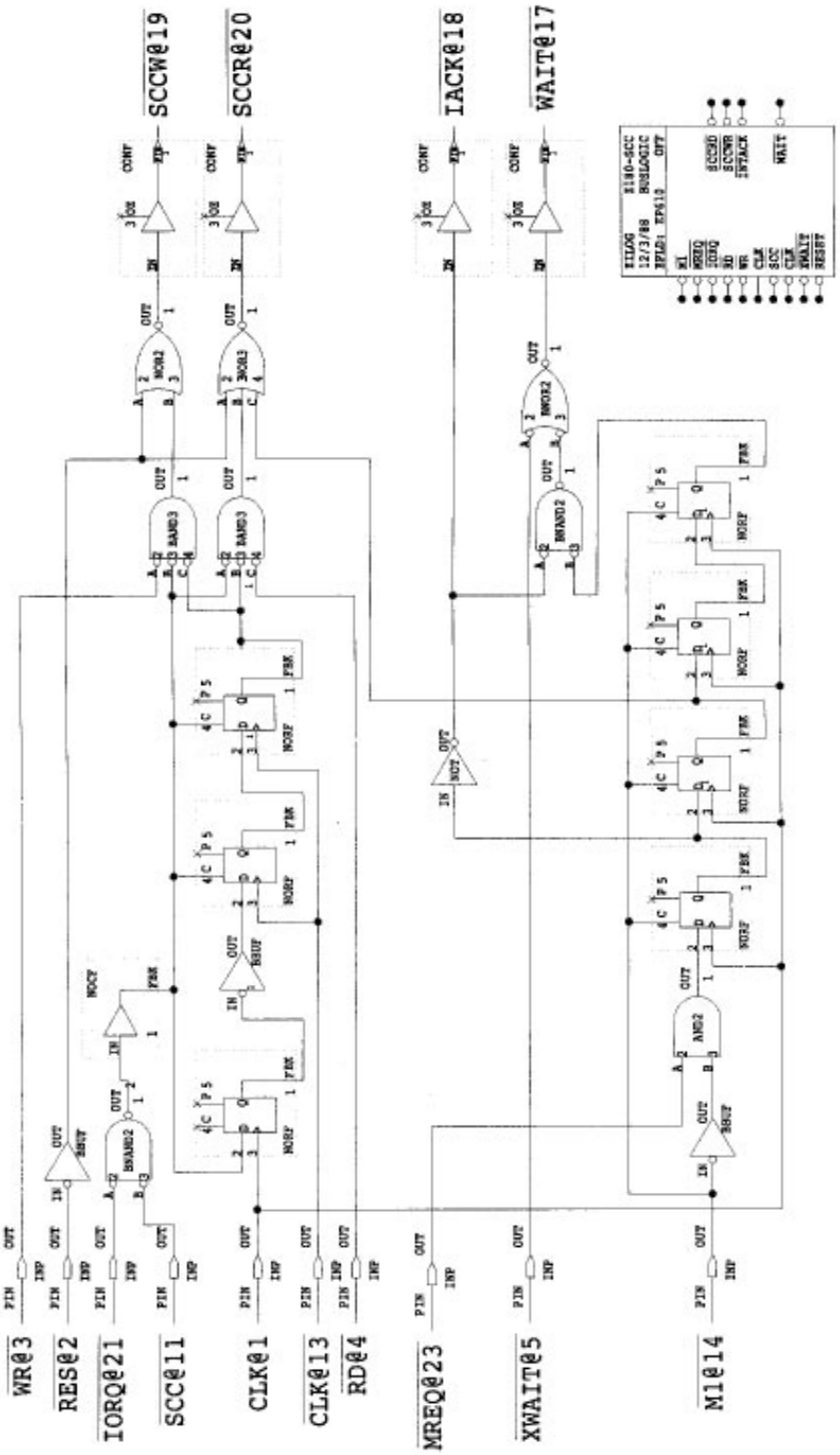



**Figure 16a. ELPD Circuit Implementation**

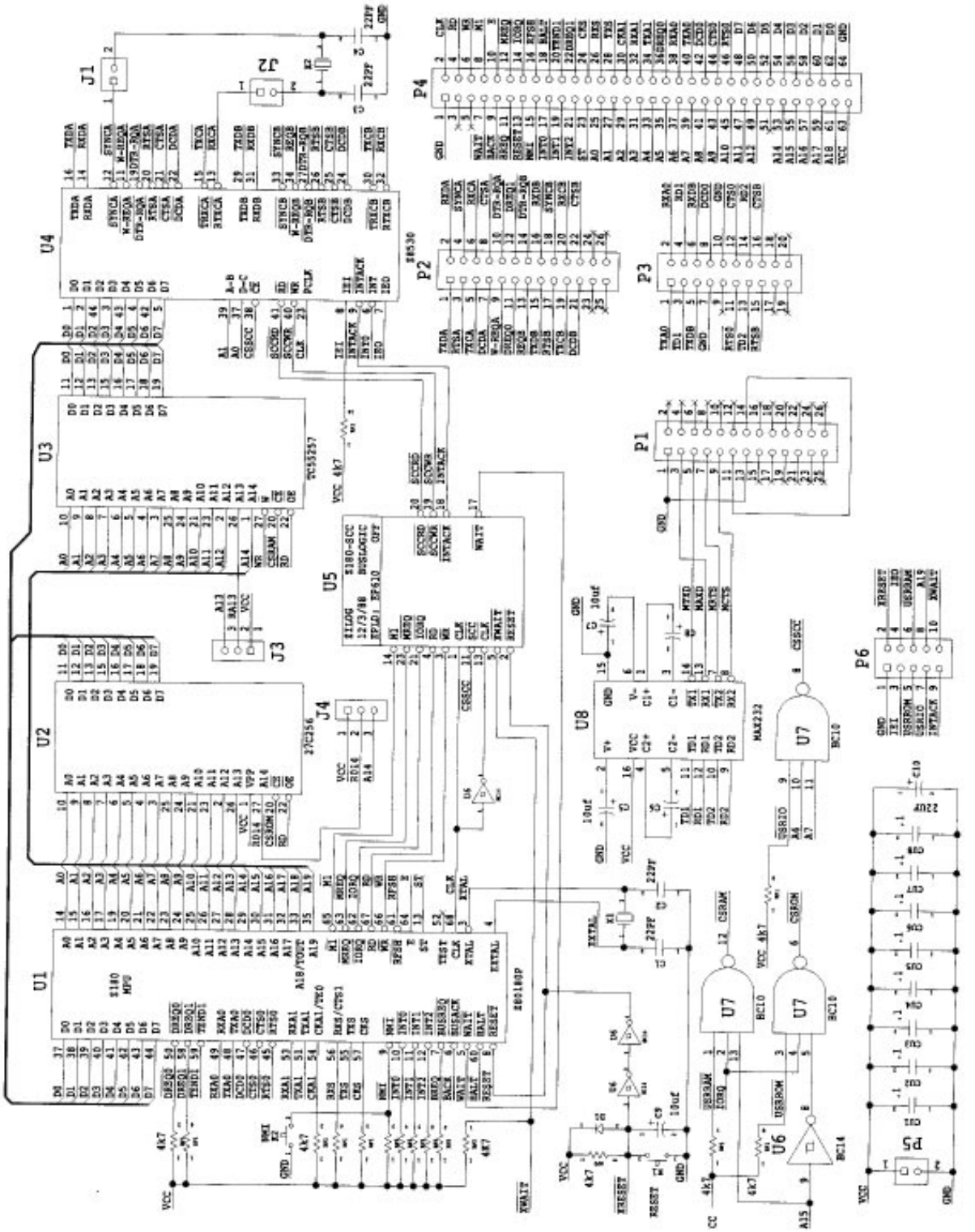


**Figure 16b. ELPD Circuit Implementation**

(Continued)

## System Checkout

After completion of the board (PC board or wire wrapped board, etc.), the following methods verify that the board is working.

## Software Considerations

Based on the previous discussion, it is necessary to program the Z180 internal registers, as follows, before system checkout:

- Z80 mode of operation - Clear /M1E bit in OMCR register to zero (to provide expansion for Z80 peripherals).
- Z80 compatible mode - Clear IOC bit in OMCR register to zero.
- Put one wait state in memory cycle, and no wait state for I/O cycle DMCR register bits 7 and 6 to “1” and bits 5 and 4 to “0”.

## SCC Read Cycle Proof

Read cycle checking is first because it is the simplest operation. The SCC Read cycle is checked by reading the bits in RR0. First, the SCC is hardware reset by simultaneously pulling /RD and /WR LOW (The circuit above includes the circuit for this). Then, reading out the Read Register 0 returns:

D7-D0 = 01xxx100b
Bit D2, D6:1
Bit D7, D1, D0:0
Bit D5: Reflects /CTS pin
Bit D4: Reflects /SYNC
Bit D3: Reflects /DCD pin

## SCC Write Cycle Proof

Write cycle checking involves writing to a register and reading back the results to the registers which return the written value. The Time Constant registers (WR12 and WR13) and External/Status Interrupt Enable register (WR15) are on the SCC.

## Interrupt Acknowledge Cycle

Checking an Interrupt Acknowledge (/INTACK) cycle consists of several steps. First, the SCC makes an Interrupt Request (/INT) to the Z180. When the processor is ready to service the interrupt, it shows an Interrupt Acknowledge (/INTACK) cycle. The SCC then puts an 8-bit vector on the bus and the Z180 uses that vector to get the correct service routine. The following test checks the simplest case.

First, load the Interrupt Vector Register (WR2) with a vector, disable the Vector Interrupt Status (VIS) and enable interrupts (IE=1, MIE=1 IEI=1). Disabling VIS guarantees only one vector on the bus. The address of the service routine corresponding to the 8-bit vector number loads the Z180 vector table, and the Z180 is under Interrupt Mode 2.

Because the user cannot set the SCC Interrupt Pending Bit (IP), setting an interrupt sequence is difficult. An interrupt is generated indirectly via the CTS pin by enabling the following explanation.

Enable interrupt by /CTS (WR15, 20h), External/Status Interrupt Enable (WR1, 01h), and Master Interrupt Enable (WR9, 08h). Any change on the /CTS pin begins the interrupt sequence. The interrupt is re-enabled by Reset External/Status Interrupt (WR0, 10h) and Reset Highest IUS (WR0, 38h).

A sample program of an SCC Interrupt Test is shown in Table 12. The following programs in Tables 12, 13, and 14 assume that the 180 is correctly initialized. Table 12 uses the Assembler for the Z80 CPU.

**Table 12. SCC Test Program – Interrupt for 180/SCC Application Board (Under Mode2 Interrupt)**

```
;*          B register returns status info:
;*          Bit D0: current /cts stat
;*          D1set: /cts int received
;*
.z800
                        ;Read in Z180 register names and
*include 180macro.lib   ;macro for Z180 new instructions

;SCC
Registers
scc_ad:                 equ     0C3h        ;addr of scc ch a - data
scc_ac:                 equ     0C2h        ;addr of scc ch a - control
scc_bd:                 equ     0C1h        ;addr of scc ch b - data
scc_bc:                 equ     0C0h        ;addr of scc ch b - control

scc_a:                  equ     000h        ;set 0ffh to test ch a
                                            ;clear 00h to test ch b.

        if      scc_a
scc_cont:               equ     scc_ac
        else
scc_cont:               equ     scc_bc
        endif

org     09000h          ;top of user ram area

inttest:                ld      sp,top_of_sp        ;init sp
                        ld      a,high sccvect and 0ffh ;init i reg
                        ld      i,a
                        im      2                   ;set interrupt mode 2
                        call    initscc             ;initialize scc
                        ld      b,0                 ;clear status
                        ei                          ;enable interrupt

wait_loop:              bit     1,b                 ;check int status
                        jr      z,wait_loop         ;if not, loop again

wait_here:              jr      $                   ;interrupt has been received
                                                    ;you can set breakpoint here!

;subroutine to initialize scc registers
;initialization table format is
;register number, then followed by the data to be written
;and the register number is 0ffh, then return

initscc:                ld      hl,scctab           ;initialize scc
init0:                  ld      a,(hl)              ;get register number
                        cp      0ffh                ;reached at the end of table?
                        ret     z                   ;yes, return.
                        out     (scc_cont),a        ;write it
                        inc     hl                  ;point to next data
                        ld      a,(hl)              ;get the data to be written
                        out     (scc_cont),a        ;write it
                        inc     hl                  ;point to next data
                        jr      init0               ;then loop
```

(Continued)

**Table 12. SCC Test Program – Interrupt for 180/SCC Application Board (Under Mode2 Interrupt) (Continued)**

```
;external/status interrupt
service routine

ext_stat:   ld      a,10h
            out     (scc_cont),a        ;reset ext/stat int
            in      a,(scc_cont)        ;read stat
            and     00100000b           ;mask off bits other than /cts
            rra                         ;shift into D0 loc
            rra
            rra
            rra
            rra
            set     1,a                 ;set interrupt flag
            ld      b,a                 ;save it
            ld      a,38h
            out     (scc_cont),a        ;reset highest ius
            ei                          ;enable int
            ret                         ;return from int

;initialization data table for scc
;table format - register number, then value for the register
;and ends with 0ffh - since scc doesn't have
;register 0ffh...

scctab:             db      09h         ;select WR9
            if      scc_a
            db      10000000b           ;ch a reset
            else
            db      01000000b           ;ch b reset
            endif
            db      0eh                 ;select WR15
            db      20h                 ;only enable /cts int

            db      01h                 ;select WR1
            db      00000001b           ;enable ext/stat int

            db      10h                 ;reset ext/stat int
            db      10h                 ;twice

            db      09h                 ;select WR9
            db      08h                 ;mie, vect not incl. stat

            db      0ffh                ;end of table

;interrupt vector table
            org     inttest + 100h
sccvect:    dw      ext_stat

            .block  100h                ;reserve area for stack
top_of_sp:
            end
```

Table 13 shows a “macro” to enable the Z180 to use the Z80 Assembler, as well as register definitions.

There is one good test to ensure proper function. Generate a data transfer between the Z180/SCC using the Z180 on-chip DMA. The SCC self loop-back test transfers data using the Z180 DMA at the highest transmission rate (Table 13).

**Table 13. Program Example – Z180 CPU Macro Instructions**

```
;*              File name - 180macro.lib
;*       Macro library for Z180 new instructions for asm800
;*
;
;Z180 System Control Registers

;ASCI Registers
cntla0:     equ     00h         ; ASCI Cont Reg A Ch0
cntla1:     equ     01h         ; ASCI Cont Reg A Ch1
cntlb0:     equ     02h         ; ASCI Cont Reg B Ch0
cntlb1:     equ     03h         ; ASCI Cont Reg B Ch1
stat0:      equ     04h         ; ASCI Stat Reg Ch0
stat1:      equ     05h         ; ASCI Stat Reg Ch1
tdr0:       equ     06h         ; ASCI Tx Data Reg Ch0
tdr1:       equ     07h         ; ASCI Tx Data Reg Ch1
rdr0:       equ     08h         ; ASCI Rx Data Reg Ch0
rdr1:       equ     09h         ; ASCI Rx Data Reg Ch1

;CSI/O Registers
cntr:       equ     0ah         ; CSI/O Cont Reg
trdr:       equ     0bh         ; CSI/O Tx/Rx Data Reg

;Timer Registers
tmdr0l:     equ     0ch         ; Timer Data Reg Ch0-low
tmdr0h:     equ     0dh         ; Timer Data Reg Ch0-high
rldr0l:     equ     0eh         ; Timer Reload Reg Ch0-low
rldr0h:     equ     0fh         ; Timer Reload Reg Ch0-high
tcr:        equ     10h         ; Timer Cont Reg
tmdr1l:     equ     14h         ; Timer Data reg Ch1-low
tmdr1h:     equ     15h         ; Timer Data Reg Ch1-high
rldr1l:     equ     16h         ; Timer Reload Reg Ch1-low
rldr1h:     equ     17h         ; Timer Reload Reg Ch1-high
frc:        equ     18h         ; Free Running Counter

;DMA Registers
sar0l:      equ     20h         ; DMA Source Addr Reg Ch0-low
sar0h:      equ     21h         ; DMA Source Addr Reg Ch0-high
sar0b:      equ     22h         ; DMA Source Addr Reg Ch0-b
dar0l:      equ     23h         ; DMA Dist Addr Reg Ch0-low
dar0h:      equ     24h         ; DMA Dist Addr Reg Ch0-high
dar0b:      equ     25h         ; DMA Dist Addr Reg Ch0-B
bcr0l:      equ     26h         ; DMA Byte Count Reg Ch0-low
bcr0h:      equ     27h         ; DMA Byte Count Reg Ch0-high
mar1l:      equ     28h         ; DMA Memory Addr Reg Ch1-low
mar1h:      equ     29h         ; DMA Memory Addr Reg Ch1-high
mar1b:      equ     2ah         ; DMA Memory Addr Reg Ch1-b
iar1l:      equ     2bh         ; DMA I/O Addr Reg Ch1-low
iar1h:      equ     2ch         ; DMA I/O Addr Reg Ch1-high
```

(Continued)

**Table 13. Program Example – Z180 CPU Macro Instructions (Continued)**

```
bcr1l:      equ     2eh             ; DMA Byte Count Reg Ch1-low
bcr1h:      equ     2fh             ; DMA Byte Count Reg Ch1-high
dstat:      equ     30h             ; DMA Stat Reg
dmode:      equ     31h             ; DMA Mode Reg
dcntl:      equ     32h             ; DMA/WAIT Control Reg

;System Control Registers
il:         equ     33h             ; INT Vector Low Reg
itc:        equ     34h             ; INT/TRAP Cont Reg
rcr:        equ     36h             ; Refresh Cont Reg
cbr         equ     38h             ; MMU Common Base Reg
bbr:        equ     39h             ; MMU Bank Base Reg
cbar:       equ     3ah             ; MMU Common/Bank Area Reg
omcr:       equ     3eh             ; Operation Mode Control Reg
icr:        equ     3fh             ; I/O Control Reg

?b          equ     0
?c          equ     1
?d          equ     2
?e          equ     3
?h          equ     4
?l          equ     5
?a          equ     7

??bc        equ     0
??de        equ     1
??hl        equ     2
??sp        equ     3

slp         macro
            db      11101101B
            db      01110110B
            endm

mlt         macro   ?r
            db      11101101B
            db      01001100B+(??&?r AND 3) SHL 4
            endm

in0         macro   ?r, ?p




out0        macro   ?p, ?r
            db      11101101B
            db      00000001B+(?&?r AND 7) SHL 3
            db      ?p
            endm

otim        macro
            db      11101101B
```

**Table 13. Program Example – Z180 CPU Macro Instructions (Continued)**

```
        db      10000011B
        endm

otimr   macro
        db      11101101B
        db      10010011B
        endm

otdm    macro
        db      11101101B
        db      10001011B
        endm

otdmr   macro
        db      11101101B
        db      10011011B
        endm

tstio   macro   ?p
        db      11101101B
        db      01110100B
        db      ?p
        endm

tst     macro   ?r
        db      11101101B
        ifidn   <?r>,<(hl)>
                db      00110100B
        else
        ifdef   ?&?r
                db      00000100B+(?&?r AND 7) SHL 3
        else
                db      01100100B
                db      ?r
        endif
        endif
        endm
        .list
end
```

(Continued)

Table 14 lists a program example for the Z180/SCC DMA transfer test.

**Table 14. Test Program – Z180/SCC DMA Transfer**

```
;
;*    Test program for 180 DMA/SCC
;*
;*    Test 180's DMA function with SCC
;*
;*    180 dma - dma0 for scc rx data
;*         dma1 for scc tx data
;*     async, X1 mode, 1 stop, speed = pclk/4
;*         self loop-back
;*     Connect W/REQ to DREQ0 of 180
;*         DTR/REQ to DREQ1 of 180
;*
;*     B register returns status info:
;*     Bit D0 set : Tx DMA end
;*         D1 set : Rx DMA end
;*         D2 set : Data doesn't match
;*

.z800

;                                                    Read in Z180 register names and
*include 180macro.lib                                ;macro for Z180 new instructions

;SCC Registers

scc_ad:                 equ     0C3h        ;addr of scc ch a - data
scc_ac:                 equ     0C2h        ;addr of scc ch a - control
scc_bd:                 equ     0C1h        ;addr of scc ch b - data
scc_bc:                 equ     0C0h        ;addr of scc ch b - control
scc_a:                  equ     00h         ;if test ch. a, set this to 0ffh
                                            ;for ch.b, set this to 00h
        if              scc_a
scc_cont:               equ     scc_ac
scc_data:               equ     scc_ad
        else
scc_cont:               equ     scc_bc
scc_data:               equ     scc_bd
        endif

length:                 equ     1000h       ;transfer length

                        org     09000h      ;top of user ram area

sccdma:                 ld      sp,tx_buff  ;init sp
                        ld      a,(high z180vect) and 0ffh  ;init i reg
                        ld      i,a
                        ld      a,00h       ;init il
                        out0    (il),a
                        im      2           ;Set interrupt mode 2
                        call    fill_mem    ;initialize tx/rx buffer area
                        call    initscc     ;initialize scc
```

**Table 14. Test Program – Z180/SCC DMA Transfer (Continued)**

```
                        call    initdma
                        ld      b,0             ;init status

                        ld      a,00h           ;load 1st data to be sent
                        out     (scc_data),a

                        ld      a,11001100b     ;enable dmac and int from DMA0
                        out0    (dstat),a

                        ld      a,05h           ;select WR5
                        out     (scc_cont),a
                        ld      a,01101000b     ;start tx
                        out     (scc_cont),a

                        ei                      ;wait here for completion

loop:                   bit     1,b             ;rx dma end?
                        jr      z,loop          ;not, then loop again

                        push    bc              ;save bc reg
                        ld      bc,length       ;compare tx data with rx data
                        ld      de,tx_buff
                        ld      hl,rx_buff
chkloop:    ld          a,(de)
                        cpi
                        jr      nz,bad_data
                        jp      v,good
                        inc     de
                        jr      chkloop
bad_data:               pop     bc              ;restore bc
                        set     2,b             ;set error flag
                        jr      enddma
good:                   pop     bc              ;restore bc

enddma:                 jr      $               ;tx/rx completed
;                                               you can put breakpoint here

fill_mem:   l           d       hl,temp         ; prepare data to be sent
                        ld      bc,length       ; set length
                        ld      de,tx_buff
                        ld      (hl),00h
fill_loop:              ldi
                        jp      nv,fill_00
                        dec     hl
                        inc     (hl)
                        jr      fill_loop

fill_00:                ld      bc,length       ; clear rx buffer area to zero
                        ld      de,rx_buff
                        ld      (hl),00h
fill_00l:               ldi
                        ret     nv
                        dec     hl
                        jr      fill_00l
```

(Continued)

**Table 14. Test Program – Z180/SCC DMA Transfer (Continued)**

```
initscc:        ld      hl,scctab       ; initialize scc
init0:          ld      a,(hl)
                cp      0ffh
                ret     z
                out     (scc_cont),a
                inc     hl
                ld      a,(hl)
                out     (scc_cont),a
                inc     hl
                jr      init0
;initialize z180's scc
;
initdma:        ld      hl,addrtab      ;initialize DMA

                ld      c,sar0l
                ld      b,dstat - sar0l
                otimr
                ld      a,00001100b     ;dmac0 - i/o to mem++
                out0    (dmode),a
                ld      a,01001000b     ;1 mem wait, no i/o wait,
                                        ;should be EDGE for Tx DMA
                                        ;NOT level
                                        ;- because of DTR/REQ timing
                ret

txend:          ld      a,00010100b     ;isr for dma1 int-complete tx
                out0    (dstat),a       ;disable dma1
                set     0,b             ;set status
                ei
                ret

rxend:          ld      a,00100000b     ;isr for dma0 int
                out0    (dstat),a       ;disable dma0
                set     1,b             ;set status
                ei
                ret

;initialization data table for scc
;table format - register number, then value for the register
;and ends with 0ffh - since scc doesn't have
;register 0ffh...
scctab:         db      09h             ;select WR9
        if scc_a
                db      10000000b       ;reset ch a
        else
                db      01000000b       ;Reset Ch B
        endif
                db      04h             ;select WR4
                db      00000100b       ;async,x1,1stop,parity off
```

**Table 14. Test Program – Z180/SCC DMA Transfer (Continued)**

```
        db      01h             ;select WR1
        db      01100000b       ;REQ on Rx

        db      02h             ;select WR2
        db      00h             ;00h as vector base

        db      03h             ;select WR3
        db      11000000b       ;Rx 8bit/char

        db      05h             ;select WR5
        db      01100000b       ;tx 8bit/char

        db      06h             ;select WR6
        db      00h             ;

        db      07h             ;select WR7
        db      00h             ;

        db      09h             ;select WR9
        db      00000001b       ;stat low, vis

        db      0ah             ;select WR10
        db      00000000b       ;set as default
        db      0bh             ;select WR11
        db      01010110b       ;
        ;       0                       No xtal
        ;        1010                   TxC,RxC from BRG
        ;           110                 TRxC = BRG output

        db      0ch             ;select WR12
        db      00h             ;BR TC Low

        db      0dh             ;select WR12
        db      00h             ;BR TC high

        db      0eh             ;select WR14
        db      00010110b       ;
        ;       000                     nothing about DPLL
        ;          1                    Local loopback
        ;           0                   No local echo
        ;            1                  DTR/REQ is req
        ;             1                 BRG source = PCLK
        ;              0                Not enabling BRG yet

        db      0eh             ;select WR14
        db      00010111b       ;
        ;       000                     nothing about DPLL
        ;          1                    Local loopback
        ;           0                   No local echo
        ;            1                  DTR/REQ is REQ
        ;             1                 BRG source = PCLK
        ;              1                Enable BRG

        db      03h             ;select WR3
        db      11000001b       ;rx enable
```

(Continued)

**Table 14. Test Program – Z180/SCC DMA Transfer (Continued)**

```
                        db      01h             ;select WR1
                        db      11100000b       ;enable DMA

                        db      0fh             ;select WR15
                        db      00000000b       ;don't use any of ext/stat int
                        db      10h             ;reset ext/stat twice
                        db      10h

                        db      01h             ;select WR1
                        db      11100000b       ;no int

                        db      09h             ;select WR9
                        db      00001001b       ;enable int

                        db      0ffh            ;end of table

;source/dist addr table for Z180's dma

addrtab:                db      scc_data        ;dmac0 source
                        db      00h
                        db      00h

                        dw      rx_buff         ;dmac0 dist
                        db      00h

                        dw      length          ;byte count

                        dw      tx_buff+1       ;mar
                        db      00h

                        db      scc_data        ;iar
                        db      00h

                        db      00h             ;dummy!

                        dw      length-1        ;byte count

;interrupt vector table

                        org     sccdma + 200h
z180vect:               .block  2               ;180 int1 vect 00000
                        .block  2               ;180 int2 vect 00010
                        .block  2               ;180 prt0 vect 00100
                        .block  2               ;180 prt1 vect 00110
                        dw      rxend           ;180 dmac0 vect 01000
                        dw      txend           ;180 dmac1 vect 01010
                        .block  2               ;180 csi/o vect 01100
                        .block  2               ;180 asci0 vect 01110
                        .block  2               ;180 asci1 vect 10000

                        org     sccdma + 1000h
tx_buff:                .block  length
rx_buff:                .block  length
temp:                   .block  1

end
```

First, this program (Table 14) initializes the SCC by:

Async, X1 mode, 8-bit 1 stop, Non-parity.
Tx and Rx clock from BRG, and BRG set to PCLK/4.Self Loopback

Then, it initializes 4K bytes of memory with a repeating pattern beginning with 00h and increases by one to FFh (uses this as Tx buffer area). Also, it begins another 4K bytes of memory as a Rx buffer with all zeros. After starting, DMA initialization follows:

**DMAC0:** For Rx data transfer: I/O to Mem, Source address- fixed, Destination address-increasing. Edge sense mode: Interrupt on end of transfer.

**DMAC1:** For Tx data transfer: Mem to I/O, Source address-increasing, Destination address - fixed. Edge sense mode: Interrupt on end of transfer.

Now, start sending with DMA.

On completion of the transfer, the Z180 DMAC1 generates an interrupt. Then, wait for the interrupt from DMAC0 which shows an end of receive. Now, compare received data with sent data. If the transfer was successful (source data matched with destination), 00h is left in the accumulator. If not successful, 0FFh is left in the accumulator.

This program example specifies a way to initialize the SCC and the Z180 DMA.

## CONCLUSION

This Application Note describes only one example of implementation, but gives you an idea of how to design the system using the Z180™ and SCC.

For further design assistance, a completed board together with the Debug/Monitor program and the listed sample program are available. If interested, please contact your local Zilog sales office.

# THE ZILOG DATACOM FAMILY WITH THE 80186 CPU

*Zilog's datacom family evaluation board features the 80186 along with four multiprotocol serial controllers, and allows customers to evaluate these components in an Intel environment.*

## INTRODUCTION

Zilog's customers need a way to evaluate its serial communications controllers with a central CPU. This App Note (Application Note) explains and illustrates how the datacom family interfaces and communicates with the 80186 on this evaluation board. The board helps the potential customer to evaluate Zilog's data communications controllers in an Intel environment.

The most advanced and complex component of the serial family is the IUSC. One of the highlights of this App Note is how the IUSC adapts to the 80186 CPU with a minimum of difficulty and a maximum of bus and functional flexibility.

## GENERAL DESCRIPTION

The evaluation board includes the following hardware. (Reference two page Schematic diagram at rear of the App Note - Figures 5A and 5B.)

- Intel 80186 Integrated 16-bit Microprocessor
- Zilog Z16C32 Integrated Universal Serial Controller (IUSC™)
- Zilog Z16C33 Monochannel Universal Serial Controller (MUSC™) or USC®
- Zilog Z16C35 Integrated Serial Communications Controller (ISCC™)
- Zilog Z85230 Enhanced Serial Communications Controller (ESCC™) or SCC
- Two 28-pin EPROM sockets, suitable for 2764's through 27512's
- Six 32-pin (or 28-pin) SRAM sockets, suitable for 32K x 8 or 128K x 8 devices
- Four Altera EPLD circuits comprising the glue logic (Figures 1-4 at rear of the App Note) and Evaluation Board Schematic (Figures 5a, 5b)
- RS-232 and RS-422 line drivers and receivers
- Pin headers for configuring and interconnecting the above to serial applications

**Notes:**
All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

Power connections follow conventional descriptions below:

| Connection | Circuit | Device |
|---|---|---|
| Power | $V_{CC}$ | $V_{DD}$ |
| Ground | GND | $V_{SS}$ |

## GENERAL DESCRIPTION (Continued)

### Processor

The 80186 may be operated at rates up to 16 MHz. To use the CPU clock for accurate serial bit clocking, a 9.8304 MHz CPU clock can be used. The crystal connected to the processor is 2X the operating frequency.

The processor's 1 Mbyte address space is well filled if the maximum RAM complement is installed. Of the integrated Chip Select outputs provided by the 80186, the /UCS output is used for the EPROMs, and all of the /PCS6-/PCS0 outputs are used for the datacom controllers. A hardware address decoder is used for the SRAMs instead of the 80186's /LCS and /MCS3-/MCS0 outputs because the RAMs must be accessible to the on-chip DMA functions of the ISCC and IUSC as well as the 80186. The 80186 does not decode addresses from external bus masters. Both 8-bit and 16-bit accesses are provided for RAM. The EPROMs are only accessible to the 80186.

The 80186's mid-range memory chip select feature (specifically, the /MCS2 output) is used to give the software a way to hardware Reset the ISCC, IUSC, and (M)USC. This allows a customer's program to operate as if it were in a target system starting from Reset, including the initial write to the Bus Configuration Register (BCR).

The 80186's two integrated DMA channels can be used for any two of the four or six serial data streams in the B side of the (E)SCC and the (M)USC. The "DMA EPLD" derives requests for the 80186's two DMA channels from six inputs, two each for (E)SCC channel B and the one or two channels in the (M)USC. It asserts DREQ0 or DREQ1 (High) if any of the inputs for that channel is low, and the 80186 is not performing an Interrupt Acknowledge cycle. Jumper blocks J22, J23, J24, and J29 control the assignment of the 80186's internal DMA controllers, including provision for a clipped Tx request that is needed if a standard SCC is installed in place of the ESCC. The various possibilities are summarized in Table 1.

**Table 1. 80186 DMA Jumper Connections**

| To enable the following to use 80186 DMA Channel 0: | Install this jumper: |
|---|---|
| (E)SCC B Rx | J23-1 to J23-2 |
| MUSC Rx or USC A Rx | J22-1 to J22-2 |
| MUSC Tx or USC A Tx | J22-4 to J22-2 |
| USC B Rx | J29-1 to J29-2 |
| USC B Tx | J29-4 to J29-2 |
| **To enable the following to use 80186 DMA Channel 1:** | **Install this Jumper:** |
| ESCC B Tx | J24-1 to J24-3 |
| (E)SCC B Tx w/early release | J24-1 to J24-2 |
| MUSC Rx or USC A Rx | J22-1 to J22-3 |
| MUSC Tx or USC A Tx | J22-4 to J22-3 |
| USC B Rx | J29-1 to J29-3 |
| USC B Tx | J29-4 to J29-3 |

If more than one channel among the ESCC B and (M)USC are enabled for one of the 80186's internal DMA channels, software must ensure that only one of the enabled devices makes requests during a given block transfer. This can be done by leaving an entire Receiver or Transmitter idle or disabled, or by programming the device so that the DMA request is not output on the pin.

The ISCC and IUSC handle their own DMA transfers via the 80186's HOLD/HLDA facility.

**Note:** Either a Z16C33 MUSC or a Z16C30 USC can be installed in socket U5. If this is done, references to the (M)USC herein after may mean the USC as a whole or just its channel A; which one should be clear from the context.

The inputs and outputs associated with the processor's integrated counter/timer facility are brought to the pin header labelled J26 so that they can be used in applications (Table 2).

**Table 2. Counter/Timer Signal Locations**

| J26 pin | Signal |
|---|---|
| 1 | Timer In 1 |
| 2 | Timer Out 1 |
| 3 | Timer In 0 |
| 4 | Timer Out 0 |
| 5 | N/C |
| 6 | Ground |

The 80186's integrated interrupt controller is largely bypassed in favor of the traditional Zilogical interrupt daisy-chain structure.

Push buttons are provided for Reset and Non-Maskable Interrupt (NMI). A means to generate an NMI, in response to a Start bit received from the user's PC or terminal, is also provided. The first transmitted Start bit on the RS-232. Console connector J1, after a Reset, also produces an NMI; this feature can be used to find which serial controller channel is connected to the Console connector.

## Address Map

EPROM is located at the highest addresses, and its size is programmable in the 80186 for the /UCS output. The addresses of the datacom controllers are programmed in the 80186 for the /PCS6-/PCS0 outputs, as a block of 128x7=896 bytes starting at a 1 Kbyte boundary. The block can be in I/O space or in a part of memory space that is not used for SRAM or EPROM. The starting 1 Kbyte boundary is called (PBA) in the following sections.

RAM extends upward from address 0.

Using 128K x 8 SRAMs and 64K x 8 EPROMs, the address map might be as shown in Table 3.

**Table 3. Suggested Address Map**

| | |
|---|---|
| RAM | 00000-BFFFF |
| (E)SCC | D8000, 2, 4, 6 or D8000-D803E (even addrs only) |
| ISCC | D8080-D80FE (even addrs only) |
| (M)USC | D8100-D81FF |
| IUSC | D8200-D837F |
| ISCC-IUSC-(M)USC Reset | DB000-DB7FF (if enabled) |
| 27512 EPROM | E0000-FFFFF |

## EPROM

Two 28-pin EPROM sockets are provided; both must be populated in order to handle the 80186's 16-bit instruction fetches. Jumper header J18 allows the sockets to be compatible with 2764s, 27128s, 27256s, or 27512s; it is jumpered at the factory to match the EPROMs provided. For 27512s only, jumper J18-J2 to J18-J3 and leave J18-J1 open. For 2764s, 27128s, or 27256s, jumper J18-J2 to J18-J1 and leave J18-J3 open.

Note: J18 connects pin 1 of both sockets to either A16 or Vcc. This is done because for 2764s, 27128s, and 27256s, pin 1 is Vpp which may require a high voltage and/or draw more current than a normal logic input. For 2764s and 27128s, a similar jumper might be provided in some designs for pin 27 (/PGM). As long as the address for /UCS is programmed as described in the next paragraph, A15 (which is connected to pin 27) is High whenever /UCS is Low, so that 2764s and 27128s operate correctly.

The first code executed after Reset should program the 80186's Chip Select Control Registers to set up the address ranges for which outputs like /UCS and /PCS6-/PCS0 are asserted. In particular, the UMCS register (address A0H within the 80186's Peripheral Control Block) must be programmed to correspond to the size of EPROMs used (Table 4).

**Table 4. EPROM Address Ranges**

| EPROM Type | UMCS Value | EPROM | Address Range |
|---|---|---|---|
| | 2764 | FC3C | FC000-FFFFF |
| | 27128 | F83C | F8000-FFFFF |
| | 27256 | F03C | F0000-FFFFF |
| | 27512 | E03C | E0000-FFFFF |

The three LSBs of the above UMCS values are all 100, which signifies no external Ready/WAIT is used and no wait states are required. If the EPROMs are not fast enough for no-wait-state operation, making the three LSBs 101, 110, or 111 extends EPROM cycles by 1, 2, or 3 wait states, respectively.

## RAM

Six 32-pin sockets are provided; they should be populated in pairs, starting with the lower-numbered sockets, to allow for 16-bit accesses. $V_{CC}$ is provided at both pin 32 and pin 30 so that 28-pin 32K x 8 SRAMs can be installed in pins 3-30 of the sockets. Jumper block J19 allows decoding of the Chip Select signals from A17-A16 for 32K x 8 SRAMs or from A19-A18 for 128K x 8 SRAMs. The six standard memory populations are:

| | |
|---|---|
| One pair of 32K x 8 devices: | 64 Kbytes at 00000-0FFFF |
| Two pairs of 32K x 8 devices: | 128 Kbytes at 00000-1FFFF |
| Three pairs of 32K x 8 devices: | 192 Kbytes at 00000-2FFFF |
| One pair of 128K x 8 devices: | 256 Kbytes at 00000-3FFFF |
| Two pairs of 128K x 8 devices: | 512 Kbytes at 00000-7FFFF |
| Three pairs of 128K x 8 devices: | 768 Kbytes at 00000-BFFFF |

J19 is factory set according to the size of the SRAMs provided. For 32K x 8 SRAMs, jumpers are installed between J19-J2 and J19-J3, and between J19-J5 and J19-J6, with J19-J1 and J19-J4 left open. For 128K x 8 SRAMs, jumpers are installed between J19-J1 and J19-J2, and between J19-J4 and J19-J5, with J19-J3 and J19-J6 left open.

32K x 8 SRAMs have cyclic/redundant addressing starting at 40000, 80000, and C0000. The only configuration in which this causes problems is with three pairs of 32K x 8 SRAMs and 27512 EPROMs; in this case, there is a conflict in the range E0000-EFFFF. This conflict can be avoided by any of the following means:

- Using two pairs of 32K x 8 SRAMs;
- Using one pair of 128K x 8 SRAMs;
- Using 27256 EPROMs, or
- Using 27512 EPROMs but programming the size of /UCS like they are 27256s.

Since the /LCS output of the 80186 is not used, the LMCS register in the 80186 is not written with any value.

**Programming the Peripheral Chip Selects**
The 80186 allows the /PCS6-/PCS0 pins, which in this case select the various datacom controllers, to be asserted for a selected 896-byte block of addresses. The block may reside in either memory or I/O space depending on the values programmed into the PACS and MPCS registers, locations A4H and A8H of the 80186's Peripheral Control Block, respectively. The choice of address space depends on the needs of the customer's application and the configuration of software supplied with the board (Table 5).

**Table 5. Three Standard Alternatives for Serial Controller Addressing**

| Basic Requirement | Base Address (PBA) | PACS value | MPCS value |
|---|---|---|---|
| I/O Space | 8000 | 0838 | 81B8 |
| Memory Space, 32K x 8 SRAMS used | 38000 | 3838 | 81F8 |
| Memory Space, 128K x 8 SRAMs used | D8000 | D838 | 81F8 |

The three LSBs of the PACS value specify the Ready/WAIT handling for the /PCS3-/PCS0 lines which select the (E)SCC, ISCC, and (M)USC. The three LSBs of the MPCS value specify the Ready/WAIT handling for the /PCS4, 5, and 6 lines, which select the IUSC. Both fields are shown here with the LSB's 000, signifying that the 80186 should honor a WAIT on the external Ready/WAIT signal, but that it should not provide any minimum wait.

**Programming the Mid-Range Memory to Reset the ISCC, IUSC, and (M)USC**
A Reset puts the ISCC, IUSC, and (M)USC in a special and unique state in which the first write to each device implicitly goes to a Bus Configuration Register (BCR) that controls the device's basic bus operation; the BCR is not accessible thereafter. So that this board can serve as a complete development environment for customers' software, it includes a means whereby software (e.g., the debug monitor) can assert the /RESET input of these three devices. Specifically, assertion of the /MCS2 output of the 80186 causes such a Reset.

The 81 in the MS Byte of the MPCS values, shown in Table 5, makes each of the /MCS3-/MCS0 pins correspond to a 2 Kbyte block of addresses in memory space. The actual active pin addresses are determined by the value written into the MMCS register; location A6H of the 80186' Peripheral Control Block. Table 6 shows suggested MMCS values as a function of the RAM chip size, and the corresponding range of addresses for which any read or write access causes the three controllers to be reset.

**Table 6. Address Ranges for Reset**

| RAM Size | MMCS value | Address Range for which ISCC, IUSC, and (M)USC are Reset: |
|---|---|---|
| 32K x 8 | 3BFF | 3B000-3B7FF |
| 128K x 8 | DBFF | DB000-DB7FF |

The three LSBs of the above MMCS values are 111 so that the longest possible Reset pulse is generated when any of the locations in the indicated range are accessed.

Note that if this feature is not needed, it can be disabled by simply not programming the MMCS register.

**Interrupt Daisy Chain (Priority) Order**

Jumper block J25 selects whether the (E)SCC device is at the start or the end of the interrupt daisy chain.

| To make the interrupt priority be: | Jumper J25 as follows: |
|---|---|
| (E)SCC highest, IUSC, ISCC, (M)USC lowest | J25-J2 to J25-J3, J25-J4 to J25-J5 (J25-J1, J25X open) |
| IUSC highest, ISCC, MUSC, (E)SCC lowest | J25-J1 to J25-J2, J25-J to J25-J4 (J25-5J, J25X open) |
| IUSC highest, ISCC, USC, (E)SCC lowest | J25X to J25-J2, J25-J3 to J25-J4 (J25-J1, J25-J5 open) |

This variability is provided in part because early versions of the 85230 ESCC had trouble passing an interrupt acknowledge down the daisy chain if it occurred in response to a lower-priority device's request just as the ESCC was starting to make its own request. Current 85230's don't have the problem.

## (E)SCC

Socket U2 can be configured for either an ESCC or SCC, and for versions thereof that use either multiplexed or non-multiplexed address and data. Jumper blocks J20 and J21 select certain signals accordingly. For a part with multiplexed addresses and data (80x30), jumper J20-J1 to J20-J2 and leave J20-J3 open, and jumper J21-J1 to J21-J2 and J21-J4 to J21-J5, leaving J21-J3 and J21-J6 open. With such a part, software can directly address the (E)SCC's registers, and need not concern itself with writing register addresses to Write Register 0 (WR0).

For a part having a non-multiplexed bus (85x30), jumper J20-J2 to J20-J3, J21-J2 to J21-J3, and J21-J5 to J21-J6, leaving J20-J1, J21-J1, and J21-J4 open. In this case, software must handle the (E)SCC by writing register addresses into its WR0 in order to access any register other than WR0, RR0, or the data registers.

Channels A and B can be handled on a polled or interrupt-driven basis. Channel A of the (E)SCC is suggested for connecting the user's PC or terminal for use with the Debug Monitor included in this evaluation kit. Channel B (but not A) can be handled on a DMA basis using the 80186's internal DMA channels, or on a polled or interrupt driven basis.

Jumper block J23 allows channel B's /W//REQB output to be used for either a Wait function or a Receive DMA Request function. To use the output for Wait, jumper J23-J2 to J23-J3 and leave J23-J1 open. The Wait function is only significant if the software wants to delay completion of a Read from the (E)SCC's Receive Data register until data is available, and/or if it wants to delay completion of a Write to the Transmit Data register until the previously-written character has been transferred to the Transmit Shift register. These modes are alternatives to checking the corresponding status flags and can be used to achieve operating speeds higher than those possible with such traditional polling, although not as fast as the speeds possible with a DMA approach.

To use the /W//REQB output as a Receive DMA Request, jumper J23-J1 to J23-J2 and leave J23-J3 open.

Jumper block J24 determines how channel B's /DTR/ /REQB output is used. To use this output for the Data Terminal Ready function, jumper J24-J3 to J24-J4 and leave J24-J1 and J24-J2 open. To use this output directly as a Transmit DMA Request (using the ESCC's early-release capability), jumper J24-J1 to J24-J3 and leave J24-J2 and J24-J4 open. To drive the Transmit DMA Request with a clipped version of this signal that is forced High earlier than a standard SCC drives it High, jumper J24-J1 to J24-J2 and leave J24-J3 and J24-J4 open.

The "SCC EPLD" handles the (E)SCC's signalling requirements. Among other things, this EPLD configures the (E)SCC socket's pins 35 and 36 for either a multiplexed or non-multiplexed part, based on whether J20 is jumpered to connect the 80186 ALE signal to one of its input pins. If the device detects high-going pulses on this input, it drives corresponding low-going Address Strobe pulses onto (E)SCC pin 35 and drives low-going Data Strobe pulses onto (E)SCC pin 36.

If the SCC EPLD's pin 9 stays at Ground, the part drives Read strobes onto pin 36 and drives delayed Write strobes onto pin 35, for a non-multiplexed 85x30 device.

While the ESCC's relaxed timing capability allows the 80186's /WR output to be connected directly to the /WR input of a non-multiplexed ESCC, the SCC EPLD delays start of an SCC's write cycle until write data is valid, even though this is not necessary for an ESCC.

The SCC EPLD also generates the clipped-DMA-request signal mentioned in connection with J24, and logically ORs Reset onto pins 35 and 36. The device also tracks the two IACK cycles provided by the 80186 for each Interrupt Acknowledge cycle. For a multiplexed address/data port, it drives the address strobe (only) on the first cycle, and it provides the /RD or /DS pulse needed by the (E)SCC (only) on the second cycle. The "DMA EPLD" provides the INTACK signal needed by the (E)SCC.

The (E)SCC is only accessible at even addresses. For a non-multiplexed part (85x30), the following four register locations are repeated throughout the even addresses from (PBA) through (PBA)+126:

| | |
|---|---|
| (PBA), (PBA)+8,... (PBA)+120 | Channel B Command/Status register |
| (PBA)+2, +10,... (PBA)+122 | Channel B Data register |
| (PBA)+4, +12, ... (PBA)+124 | Channel A Command/Status register |
| (PBA)+6, +14, ... (PBA)+126 | Channel A Data register |

For a multiplexed part (80 x 30), the Select Shift Left command (D1-0=11) should be written to Channel B's WR0 before any other registers are accessed. Then the basic (E)SCC register map occurs twice in the even addresses from (PBA) through (PBA)+126:

| | |
|---|---|
| (PBA), (PBA)+2, ... (PBA)+30 | Channel B registers 0-15 |
| (PBA)+32, +34, ... (PBA)+62 | Channel A registers 0-15 |
| (PBA)+64, +66, ... (PBA)+94 | Channel B registers 0-15 |
| (PBA)+96, +98, ... (PBA)+126 | Channel A registers 0-15 |

The redundant addressing of the (E)SCC is used to control a feature that can be used by software to allow the user to interrupt software execution from his keyboard. If the (E)SCC is read at an address with A6-A5=11 (for a multiplexed part this means in the higher-addressed A channel), a mode is set in which a low on the console Received Data line (i.e., a Start bit on pin 3 of the J1 connector) causes a Non-Maskable Interrupt on the 80186. The mode is cleared by Reset, or when the (E)SCC is read at an address with A6-A5=10 (on a multiplexed part, in the higher-addressed B channel). The NMI handler should do the latter fairly quickly to prevent subsequent data bits on Received Data from causing further NMIs.

## ISCC

Since the 80186 processor provides multiplexed addresses and data, the ISCC is configured to use the addresses on the AD lines. Therefore, software can address the various ISCC registers directly, and need not be concerned with writing register addresses into the indirect address fields of the ISCC's WR0 and CCAR.

Because the ISCC includes four DMA channels, its Channel A and B Transmitters and Receivers can be handled on a polled, interrupt-driven, and/or DMA basis, in any mixture.

Since the ISCC can only be programmed as an 8-bit device on the AD7-AD0 lines, it occupies only the even-addressed bytes within its address range, (PBA)+128 through (PBA)+254.

The first write to this address range, after a Reset, implicitly writes the ISCC's Bus Configuration Register (BCR). To match up with the rest of the board's hardware, this first write should be a byte write that stores the hexadecimal value C6 in any even address in the first half of the ISCC's address range [(PBA)+128 through (PBA)+190]. Details of this transaction are as follows:

- The High induced by a pull-up resistor on the ISCC's A/B input selects the WAIT protocol on the /WAIT//RDY pin, which corresponds to how the 80186 works. (In subsequent register accesses, the A/B selection is taken from A5 of the multiplexed address.)
- A Low on the ISCC's SCC//DMA input, which is connected to A6, is required by the internal logic of the ISCC. This is why the BCR write is restricted to the first half of the ISCC's address range.
- As with all transactions between the 80186 and ISCC, the address must be even because the ISCC only accepts slave-mode data on the AD7-AD0 pins.
- The MSB of the data (D7) is 1 to enable the Byte Swap feature, so that when the ISCC's DMA controller is reading transmit data from RAM, it takes alternate bytes from AD7-AD0 and AD15-AD8.
- D6 of the data is 1 so that when the ISCC's DMA controller is reading transmit data from RAM, it takes even-addressed bytes from D7-D0 and odd-addressed bytes from D15-D8 (same function as the 80186).
- D2-D1 of the data are 11 to select double-pulsed mode for the ISCC's /INTACK input. Again, this is how the 80186 works.
- D0 of the data is 0 to select Shift Left Address mode so that the ISCC subsequently takes register addressing from the AD5-AD1 lines rather than from AD4-AD0. This is because the 80186 is a 16-bit processor that locates even-addressed bytes on AD7-AD0 and odd-addressed bytes on AD15-AD8, but the ISCC only accepts slave-mode writes on the AD7-AD0 pins.

- The fact that the ISCC's internal logic sees activity on its /AS pin, which is inverted from the 80186' ALE signal, automatically conditions it for a multiplexed Address/Data bus.

Given that the BCR is written as above, the ISCC's slave-mode address map is as follows:

| | |
|---|---|
| (PBA)+128, 130, ..., (PBA)+190 | DMA Controller Registers |
| (PBA)+192, 194, ..., (PBA)+222 | ISCC Serial Channel B registers 0-15 |
| (PBA)+224, 226, ..., (PBA)+254 | ISCC Serial Channel A registers 0-15 |

## (M)USC

Since the 80186 processor provides multiplexed addresses and data, the (M)USC is configured to use the addresses on the AD lines. Therefore, the software need not write register addresses into the indirect address field of the (M)USC's CCAR.

The (M)USC's Transmitter and Receiver can be handled on a polled or interrupt-driven basis. In addition, any two of the Receivers and Transmitters in the (M)USC and Channel B of the (E)SCC can be handled on a DMA basis, using the 80186's integrated DMA controllers.

Jumper block J22 connects the (M)USC's /RxREQ and /TxREQ outputs to the "DMA EPLD" that makes the DMA Requests to the 80186. As shipped from the factory, jumpers are installed between J22-J1 and J22-J2, and between J22-J3 and J22-J4. In this configuration, the (M)USC's /RxREQ drives the 80186 DREQ0, and (M)USC /TxREQ drives the 80186 DREQ1. To reverse this assignment, jumper J22-J1 to J22-J3 and J22-J2 to J22-J4. To disconnect the (M)USC from one or both of the 80186's DMA channels, remove one or both jumpers (put them in a safe place in case you change your mind). Jumper block J29 provides the same connection-variability for the /RxREQ and /TxREQ outputs of Channel B of a USC.

Since the 80186's DMA channels are not capable of fly-by operation, the (M)USC's /RxACK and /TxACK pins have no dedicated function. They can be used for Request to Send and Data Terminal Ready; the two signals are lightly pulled up since they are not driven after Reset.

The (M)USC can be programmed using 16-bit data on the AD15-AD0 lines or 8-bit data on AD15-AD8 and AD7-AD0. It makes the distinction between 8-bit and 16-bit operations as part of its address map rather than through a control input. The PS pin of an MUSC, or the A//B pin of a USC, is connected to a latched version of 80186 A7. The D//C pin of the (M)USC is grounded. The overall address range of the (M)USC is 256 bytes, between (PBA)+256 and (PBA)+511.

The first write to this address range, after a Reset, implicitly writes the (M)USC's Bus Configuration Register (BCR). To match the rest of the board's hardware, this first write should be a 16-bit write, storing the hex value 0007 at any address in the second half of the (M)USC's range [any address in (PBA)+384 through 510, i.e., in the A channel of a USC]. Details of this transaction are as follows:

- The High on the PS or A//B input, which is connected to A7, selects the WAIT protocol on the /WAIT//RDY pin, corresponding to how the 80186 works.
- The MSB of the data (D15) is 0 because a separate non-multiplexed address is not wired to pins AD13:8 of the (M)USC.
- Bits 14-3 are required to be all zeros by the (M)USC's internal logic.
- D2 of the data is 1 to tell the (M)USC that the data bus is 16 bits wide.
- D1 of the data is 1 to select double-pulsed mode for the (M)USC's /INTACK input. This is how the 80186 works.
- D0 of the data is 1 to select Shift Right Address mode so that the (M)USC subsequently takes register addressing from the AD6-AD0 lines rather than from AD7-AD1.
- The fact that the (M)USC's internal logic sees activity on its /AS pin, which is inverted from the 80186' ALE signal, automatically conditions it for a multiplexed Address/Data bus.

Given that the BCR is written as above, the (M)USC address map is as follows:

| Starting Addr | Ending Addr | Registers Accessed |
|---|---|---|
| (PBA)+256 | (PBA)+319 | 16-bit access to MUSC regs or USC channel B regs |
| (PBA)+320 | (PBA)+383 | 8-bit access to MUSC regs or USC channel B regs |
| (PBA)+384 | (PBA)+447 | 16-bit access to MUSC regs or USC channel A regs |
| (PBA)+448 | (PBA)+511 | 8-bit access to MUSC regs or USC channel A regs |

**Note:** To maximize compatibility, program an MUSC using the second half of this range, (PBA)+384 through (PBA)+511.

While the ESCC and ISCC can drive their Baud Rate Generators from their PCLK inputs, the (M)USC has no such input. The 80186 clock output SYSCLK is brought to pins 7 of J9, J10, and J12, at which point it can be jumpered to pin 9 or 8 so that it is routed to the /TxC or /RxC pin of the device.

## IUSC

Since the 80186 processor provides multiplexed addresses and data on the AD lines, the IUSC is configured to use these addresses. Software need not write register addresses into the indirect address fields of the IUSC's CCAR and DCAR.

The IUSC's two DMA channels allow its Receiver and Transmitter to be handled on a polled, interrupt-driven, or DMA basis, in any combination.

The IUSC can be programmed using 16-bit data on the AD15-AD0 lines or 8-bit data on AD15-AD8 and AD7-AD0. The distinction between 8-bit and 16-bit operations is made as part of the address map rather than via a control input. The D//C pin of the IUSC is driven from A7 during slave cycles, and the S//D pin is driven from A8. The overall address range of the IUSC is 384 bytes from (PBA)+512 through (PBA)+895.

The first write to this address range, after a Reset, implicitly writes the IUSC's Bus Configuration Register (BCR). To match up with the rest of the board's hardware, this first write is a 16-bit write, storing the recommended hex value 00F7 at any word address in the range (PBA)+768 through (PBA)+830. Details of this transaction are as follows:

- The High on the IUSC's S//D input, which is connected to A8, selects the WAIT protocol on the /WAIT//RDY pin, which is how the 80186 works.
- It may not be required for this initial write, but it is good programming form for A6 to be zero since this is a word write. This and the previous point determine the recommended address range.
- The MSB of the data (D15) is 0 because a separate non-multiplexed address is not wired to pins AD13:8 of the IUSC.
- Bits 14-8 are more or less required to be all 0 by the IUSC's internal logic.
- D7-D6 are 11 to allow the DMA controllers to do either 16-bit transfers, or alternating byte transfers on AD7-AD0 for even-addressed bytes and on AD15-AD8 for odd-addressed bytes. This is compatible with 80186 byte ordering.
- D5-D4 of the data are 11 to select double-pulsed mode for the IUSC's /INTACK input. Again, this is how the 80186 works.
- D3 of the data is 0 to select open-drain mode on the IUSC's /BUSREQ pin. The board's control logic also drives this signal low when the ISCC asserts its Bus Request output.
- D2 of the data is 1 to tell the IUSC that the data bus is 16 bits wide.
- D1 of the data is 1 to select open-drain mode on the IUSC's /INT pin which is OR-tied with the interrupt request from the (E)SCC.
- D0 of the data is 1 to select Shift Right Address mode, so that the IUSC subsequently takes register addressing from the AD6-AD0 lines rather than from AD7-AD1.
- The fact that the IUSC's internal logic sees activity on its /AS pin, which is inverted from the 80186' ALE signal, automatically conditions it for a multiplexed Address/Data bus.

Given that the BCR is written as above, the IUSC slave-mode address map is as follows:

**IUSC** (Continued)

| Starting Addr | Ending Addr | Registers Accessed |
|---|---|---|
| (PBA)+512 | (PBA)+575 | 16-bit access to IUSC Transmit DMA registers |
| (PBA)+576 | (PBA)+639 | 8-bit access to IUSC Transmit DMA registers |
| (PBA)+640 | (PBA)+703 | 16-bit access to IUSC Receive DMA registers |
| (PBA)+704 | (PBA)+767 | 8-bit access to IUSC Receive DMA registers |
| (PBA)+768 | (PBA)+831 | 16-bit access to IUSC Serial Controller registers |
| (PBA)+832 | (PBA)+895 | 8-bit access to IUSC Serial Controller registers |

While the ESCC and ISCC can drive their Baud Rate Generators from their PCLK inputs, the IUSC cannot do this from its CLK input. The 80186 clock output SYSCLK is brought to pins 7 of J9, J10, and J12 at which point it can be jumpered to pin 9 or 8 so that it is routed to the /TxC or /RxC pin of the device.

Since the IUSC contains its own DMA channels, its /RxREQ and /TxREQ pins have no dedicated function. They can be used for Request to Send and Data Terminal Ready; the two signals are lightly pulled up to allow for the fact that they are not driven after Reset.

## SERIAL INTERFACING

The serial I/O pins of the four serial controllers are connected to the six connector blocks labelled J5 through J10. In addition, the port pins of the IUSC are connected to the J11 connector block, and the port pins of an MUSC or the B channel of a USC are connected to J12. These connector blocks can be interconnected for communication between on-board serial controllers, or they can be connected to the user's custom communications hardware on another board. As a third option, they can be connected to three on-board serial interfaces via the connector blocks labelled J13 through J15.

Two of the on-board serial interfaces use EIA-RS-232 signal levels and pin arrangement. 25-pin D connectors J1A or J2A are configured as DTE, while J1B and J2B are configured as DCE. These serial interfaces are used by connecting one of J5-J10 to J13 or J14, respectively. J1B is typically used for connection to the user's PC or terminal.

The third on-board serial interface uses EIA-422 signal levels on connector J3A,J3B, or J4, and is used by connecting one of J5-J10 to J15. The 25-pin D connector J3A uses the DTE pin arrangement put forth in the EIA-530 standard. J3B is a DCE version of EIA-530, while the 8-pin circular DIN connector, J4, is compatible with the Apple Macintosh Plus and later Macintoshes, and thus with AppleTalk/LocalTalk equipment.

The serial interface connectors are summarized in the following tables:

**Table 7. Controller Port Connectors**

| To use the following serial controller channel with off-board or on-board serial hardware: | Connect to this (these) 10-pin connector block(s): |
|---|---|
| (E)SCC Channel A | J5 |
| (E)SCC Channel B | J6 |
| ISCC Channel A | J7 |
| ISCC Channel B | J8 |
| IUSC | J9 (J11 for Port pins) |
| (M)USC | J10 (J12 for MUSC Port pins or USC channel B) |

**Table 8. On-Board Line Driver/Receiver Connectors**

| To use a serial chip controller with the following on-chip serial interface: | Connect the connector(s) from the previous table to: |
|---|---|
| J1A or J1B EIA-RS-232 Console | J13 |
| J2A or J2B EIA-RS-232 | J14 |
| RS-422 differential: J3A or J3B EIA-530 or J4 Circular-8 (DIN) | J15 |

The pin-out of the J5-J10 connectors is fairly consistent, but of necessity not identical because of differences among the various serial controllers:

**Table 9. Pin Assignments of Standard Controller Connectors**

| Pin# | J5: (E)SCC A pin | J6: (E)SCC B pin | J7,8: ISCC pin | J9: IUSC pin | J10: MUSC or USC A pin | J12: USC B pin |
|---|---|---|---|---|---|---|
| 1 | TxD | TxD | TxD | TxD | TxD | TxD |
| 2 | RxD | RxD | RxD | RxD | RxD | RxD |
| 3 | /RTS | /RTS | /RTS | (N/C) | /RxACK | /RxACK |
| 4 | /CTS | /CTS | /CTS | /CTS | /CTS | /CTS |
| 5 | /DTR | /DTR or (N/C) [1] | /DTR | (N/C) | /TxACK | /TxACK |
| 6 | /DCD | /DCD | /DCD | /DCD | /DCD | /DCD |
| 7 | /SYNC | /SYNC | /SYNC | (SYSCLK) | (SYSCLK) | (SYSCLK) |
| 8 | /RTxC | /RTxC | /RTxC | /RxC | /RxC | /RxC |
| 9 | /TRxC | /TRxC | /TRxC | /TxC | /TxC | /TxC |
| 10 | GND | GND | GND | GND | GND | GND |
| 11 | NA | NA | NA | /TxREQ | /TxREQ | /TxREQ |
| 12 | NA | NA | NA | /RxREQ | /RxREQ | /RxREQ |

**Note:**
[1] Controlled by the J24 jumper block: must be N/C if (E)SCC channel B transmitter is to be handled by an 80186 DMA channel.

The ground pins are included as signal references with off-board hardware.

When interconnecting between two connectors among J5-J10, DO NOT jumper corresponding pins straight across, as this connects outputs to outputs and inputs to inputs. Rather, connect at least each pin 1 to the other pin 2, and enough opposing inputs and outputs as needed to make the communication protocol meaningful.

The pin-out of the 12-pin J13-J15 connectors is similar to that of J5-J10, but more extensive. To allow for the “DCE” connectors that were added in revision “B” of the board, J13 and J14 are 16-pin headers and J15 is a 14-pin one:

**Table 10. Pin Assignments of Line Driver/Receiver Connectors**

| Pin # | J13-J14 DTE signal | J13-J14 DCE signal | J15 DTE signal | J15 DCE signal | Direction/where used |
|---|---|---|---|---|---|
| 1 | TxD | RxD | TxD | RxD | Output to J1-J4 |
| 2 | RxD | TxD | RxD | TxD | Input from J1-J4 |
| 3 | /RTS | /CTS | /RTS | /CTS | Output to J1-J3 |
| 4 | /CTS | /RTS | /CTS | /RTS | Input from J1-J4 [3] |
| 5 | /DTR | /DSR | /DTR | /DSR | Output to J1-J4 |
| 6 | /DSR | /DTR | /DSR | /DTR | Input from J1-J4 |

**Note:**
[3] Various conventions have been used to combine synchronous clock inputs and modem control inputs on Apple Macintosh connectors similar to J4, as described in a later section.

## SERIAL INTERFACING (Continued)

**Table 10. Pin Assignments of Line Driver/Receiver Connectors**

| Pin # | J13-J14 DTE signal | J13-J14 DCE signal | J15 DTE signal | J15 DCE signal | Direction/where used |
|---|---|---|---|---|---|
| 7 | | /DCD | | /DCD | Output to J1B, J2B, J3B |
| 8 | /DCD | | /DDC | | Input from J1A, J2A, J3A, J4 |
| 9 | | | | | |
| 10 | GND | GND | GND | GND | |
| 11 | | /RxC | | /RxC | Output to J1B, J2B, J3B |
| 12 | /RxC | | /RxC | | Input from J1A, J2A, J3A |
| 13 | /TxCO | /TxCI | /TxCO | /TxCI | Output to J1-3 |
| 14 | /TxCI | /TxCO | /TxCI | /TxCO | Input from J1-3 [3] |
| 15 | | /RI | | | Output to J1B, J2B |
| 16 | /RI | | | | Input from J1A, J2A |

**Note:**
[3] Various conventions have been used to combine synchronous clock inputs and modem control inputs on Apple Macintosh connectors similar to J4, as described in a later section.

Comparison of the two preceding charts leads to several conclusions:

- Pins 1-5 can always be jumpered straight across from a J5-J10 connector block to a J13-J15 connector block.
- In a synchronous environment, the Transmit clock can be either driven or received and the Receive clock can be received from the DTE connector or sent on the DCE connector.

The 10-pin J11 and J12 jumper blocks provide for connections to the Port pins of the IUSC and (M)USC, respectively. As with J5-J10, these connections may be to the customer's off-board custom circuits and/or to certain pins in the J13-J15 blocks. The following pin assignment is determined so that if a 2-channel USC is plugged into the (M)USC socket, J12 has the same pin-out for the USC's B channel as do J5-J10 for other channels.

**Table 11. Pin Assignments of Controller Port Connectors**

| Pin # | J11: IUSC Signal | J12: (M)USC Signal |
|---|---|---|
| 1 | PORT1 (Clock 1 In) | PORT1 |
| 2 | PORT4 (Xmit TSA Gate Out) | PORT4 (Xmit TSA Gate Out) |
| 3 | (N/C) | (N/C) |
| 4 | PORT0 (Clock 0 In) | PORT0 |
| 5 | (N/C) | (N/C) |
| 6 | PORT3 (Rcv TSA Gate Out) | PORT3 (Rcv TSA Gate Out) |
| 7 | (N/C) | (SYSCLK) |
| 8 | PORT5 (Rcv Sync Out) | PORT5 (Rcv Sync Out) |
| 9 | PORT2 | PORT2 |
| 10 | GND | GND |
| 11 | PORT6 (Rcv Sync In) | PORT6 (Rcv Sync In) |
| 12 | PORT7 (Xmit Complete Out) | PORT7 (Xmit Complete Out) |

Finally, an unpopulated 4-pin oscillator socket is included on the board with its output connected to a single jumper/wire-wrap pin. This socket can be populated with a user-supplied oscillator and connected to various clock pin(s) among J5-J15.

## Sensing which Serial Controller Channel is connected to the Console

In order to use the software provided with this evaluation board, one of the serial controller channels must be connected to a Personal Computer (or a dumb terminal) via the J1 and J13 connectors. Some versions of this software may restrict the choice to (E)SCC Channel A or the (M)USC, depending on the user's applications needs, but there is nothing in the hardware that limits the choice of which serial channel is used for the Console. However, on the J1-J4 (J13-J15) side there are two things that are special about the J1/J13 section as compared to the others. One is the provision for a Non-Maskable Interrupt in response to a received Start bit, as described earlier in the section on (E)SCC addressing.

Software can use the other special feature of the J1/J13 section, after a Reset, to sense which serial channel is connected to the Console port. A Reset signal (from power-on or the Reset button, but not from the Reset-the-ISCC, etc., address decode as described earlier) puts the "NMI" EPLD in a special mode wherein the first Start bit on the Console's Transmit Data lead causes an NMI. This feature can be used in a start-up procedure like the following, to tell which serial controller channel is used for the Console:

For each serial controller channel that the software can use for the Console:

1. Initialize the channel.
2. Send a NUL character to the channel.
3. Wait a short time to see if an NMI occurs. If so, the current channel is the Console. If not, go on to the next serial channel and try again.

If none of the allowed serial channels produces an NMI, the user has not properly jumpered any J5-J10 connector block to the J13 block.

Basic software should use the serial controller channel for the Console in a very basic, polled way. Because of this and because of similarities between the (E)SCC and the ISCC, and between the (M)USC and the IUSC, note that software allows the Console to be connected to either the (E)SCC channel A or to the (M)USC; in fact, it includes most of the code necessary to use any of the six serial controller channels for the Console.

## Notes on J4/Macintosh/AppleTalk/LocalTalk

The J4 connector is similar to that offered on various Macintosh systems. The ESCC and ISCC are particularly well adapted for use with this port, and development of USC family capability for AppleTalk/LocalTalk is of interest.

The J3 and J4 connectors cannot be used simultaneously. The J16 jumper block controls whether the RS-422 driver for Transmit Data is turned "on" and "off" under control of the associated Request to Send signal, as on the Mac, or is "on" full time, which is more suitable for the use of J3. To put the TxD driver under control of RTS, jumper J16-1 to J16-J2 and leave J16-J3 open. For full-time drive on TxD (and also the J3 RTS pins), jumper J16-J2 to J16-J3 and leave J16-J1 open.

The J17 jumper block controls whether the reception of Data Carrier Detect and Clear to Send is differential (on J3) or unbalanced, as on J4. To use differential signalling from J3, remove all jumpers from J17.

On the initial Macintosh and subsequent ones as well, Apple did the unbalanced signalling backward from standard RS-423 and RS-232 polarity for the CTS lead (also called HSK and HSKI). If you are developing code for Macintosh hardware, you can preserve Mac compatibility by jumpering J17-J3 to J17-J5 and J17-J4 to J17-J6. This grounds the CTS- lead and connects the CTS+ lead to J4-J2. It also (assuming a standard source at the other end) inverts CTS to the opposite sense from that expected by the serial controller for functions such as auto-enabling. To make the CTS input of the serial controller have its normal (low-true) sense, jumper J17-J3 to J17-J4, and J17-J5 to J17-J6– this grounds the CTS+ lead and connects the CTS- lead to J4-J2.

The DTR (HSKO) output is provided in Apple systems from Mac Plus onward and has standard RS-423 (and RS-232) polarity.

The DCD input on J4-J7 is provided in Apple systems from the Mac II and SE onward, and also has standard polarity on Apple hardware. Jumper J17-J1 to J17-J2 to ground the "+" input of the receiver; the "–" lead is connected to J4-J7.

## **SERIAL INTERFACING** (Continued)

With jumpers installed to make DCD and CTS unbalanced, J4 can also be used for an additional RS-232 serial link. Connect a “Mac to Hayes modem” cable to J4, and optionally a null modem interconnect module to the other end. The cable internally grounds the RxD+ and TxD+ leads so that RxD– and TxD– act like RS-232 signals.

Macintosh systems also include provisions for synchronous clock inputs. It is not known whether these features are used by any applications, or attached hardware. On all known Macs, the SCC’s TRxC pin is driven from the same signal as CTS; to be compatible with this feature, connect J15-J4 to pins 4 and 9 of the selected connector among J5-J10.

On the Mac SE, Mac II, and later models, a multiplexing scheme is provided on SCC channel A’s RTxC pin to drive from either the same signal as DCD, or from an on-board 3.672 MHz clock. (Channel B always had the 3.672 MHz clock.) The former capability can be provided by connecting J15-J6 to pins 6 and 8 of the selected connector among J5-J10. The latter capability can be only approximated using the 80186 clock with different baud rate divisors, or by using another oscillator. (The board includes an unpopulated 4-pin oscillator socket that might be useful in this regard.)

## JUMPER SUMMARY

Table 12 includes only those connector blocks intended to be populated by 2-pin option jumpers. J1-J15 and J26 are actual connectors meant for use with cables, jumper wires, or wire-wrapped connections.

**Table 12. Two-Pin Option Jumpers**

| Jumpers | Installed | Open |
|---|---|---|
| J9-J7 thru -9 | 7 to 8: 80186 SYSCLK is IUSC /RxC<br>7 to 9: 80186 SYSCLK is IUSC /TxC | 8: Something else on /RxC, or N/C<br>9: Something else on /TxC, or N/C |
| J10-J7 thru -9 | 7 to 8: 80186 SYSCLK is MUSC (USC A) /RxC<br>7 to 9: 80186 SYSCLK is MUSC (USC A) /TxC | 8: Something else on /RxC, or N/C<br>9: Something else on /TxC, or N/C |
| J12-J7 thru -9 | 7 to 8: 80186 SYSCLK is USC B /RxC<br>7 to 9: 80186 SYSCLK is USC B /TxC | 8: Something else on /RxC, or N/C<br>9: Something else on /TxC, or N/C |
| J16-J1 thru -3 | 1 to 2: J3, J4 TxD driven when RTS<br>2 to 3: J3, J4 TxD, RTS driven full-time | Must install one or the other |
| J17-J1 to -2<br>J17-J3 thru -6 | Unbalanced DCD- on J3 or J4<br>3 to 5 and 4 to 6: CTS+ on J4-J2<br>3 to 4 and 5 to 6: CTS- on J3 or J4 | Differential DCD+, DCD- on J3<br>Differential CTS+, CTS- on J3 |
| J18-J1 thru -3 | 1 to 2: 2764, 27128, 27256 EPROMs<br>2 to 3: 27512 EPROMs | Must install one or the other |
| J19-J1 thru -6 | 1 to 2 and 4 to 5: 128K x 8 SRAMs<br>2 to 3 and 5 to 6: 32K x 8 SRAMs | Must install one way or the other |
| J20-J1 thru -3 | 1 to 2: U2 contains 80C30 or 80230<br>2 to 3: U2 contains 85C30 or 85230 | Must install one way or the other |
| J21-J1 thru -6 | 1 to 2 and 4 to 5: U2 contains 80C30 or 80230<br>2 to 3 and 5 to 6: U2 contains 85C30 or 85230 | Must install one way or the other |
| J22-J1 thru -4 | 1 to 2: MUSC (USC A) RxREQ on DMA 0<br>1 to 3: MUSC (USC A) RxREQ on DMA 1<br>2 to 4: MUSC (USC A) TxREQ on DMA 0<br>3 to 4: MUSC (USC A) TxREQ on DMA 1 | <br>1: MUSC (USC A) Rx no DMA<br>4: MUSC (USC A) Tx no DMA |
| J23-J1 thru -3 | 1 to 2: (E)SCC B RxRQ on DMA 0<br>2 to 3: (E)SCC B Wait function | (E)SCC B neither Rx DMA<br>nor Wait |
| J24-J1 thru -4 | 1 to 2: clipped SCC B TxREQ on DMA 1<br>1 to 3: direct ESCC B TxREQ on DMA 1<br>3 to 4: /DTR output from ESCC B | (E)SCC B neither Tx DMA<br>nor /DTR |
| J25-J1 thru - 5 and J25X | 1 to 2 and 3 to 4: (E)SCC last on IACK chain, MUSC second to last<br>J25X to 2 and 3 to 4: (E)SCC last, USC 2nd to last<br>2 to 3 and 4 to 5: (E)SCC first on IACK chain | Must be one of these three ways |
| J28-J1 thru -6 | 1 to 2: 80186 SYSCLK is (E)SCC PCLK<br>3 to 4: 80186 SYSCLK is ISCC PCLK<br>5 to 6: 80186 SYSCLK is IUSC CLK | Connect some other clock to 2, 4, or 6 |
| J29-J1 thru -4 | 1 to 2: USC B RxREQ on DMA 0<br>1 to 3: USC B RxREQ on DMA 1<br>2 to 4: USC B TxREQ on DMA 0<br>3 to 4: USC B TxREQ on DMA 1 | <br>1: USC B Rx no DMA<br>4: USC B Tx no DMA |

## DMA/EPLD LOGIC

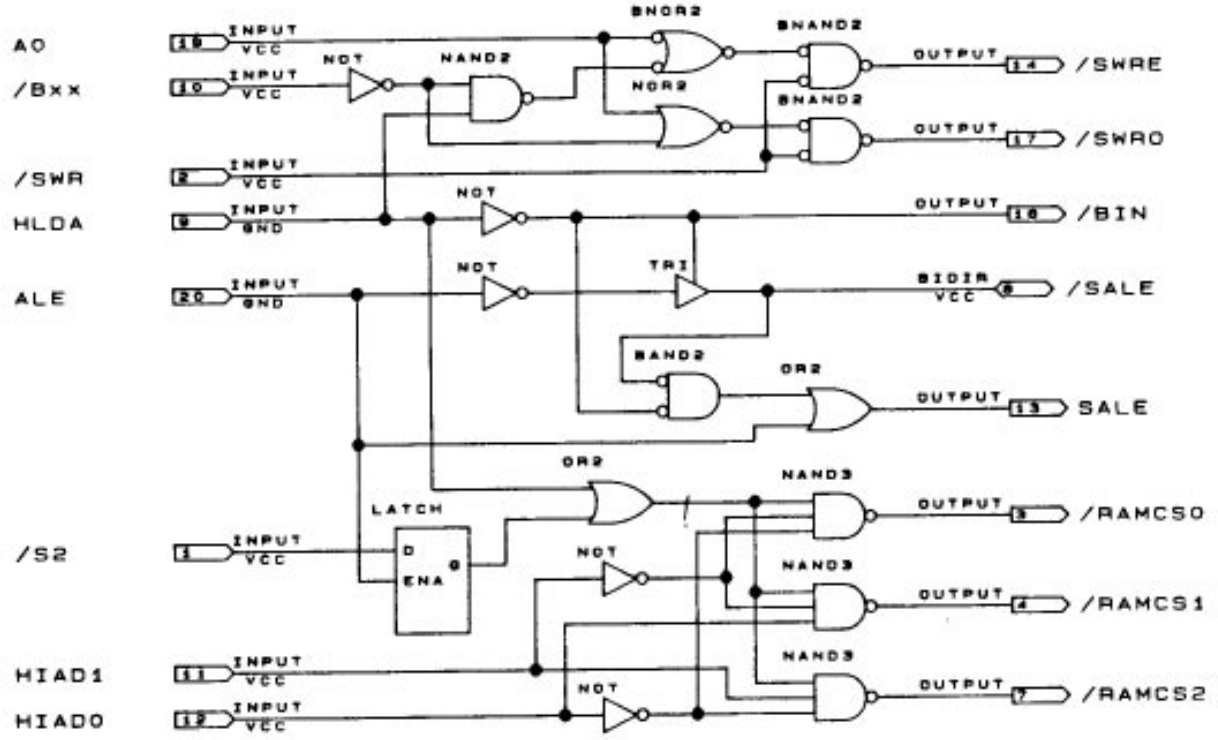



**Figure 1. Control EPLD for 186 Board**

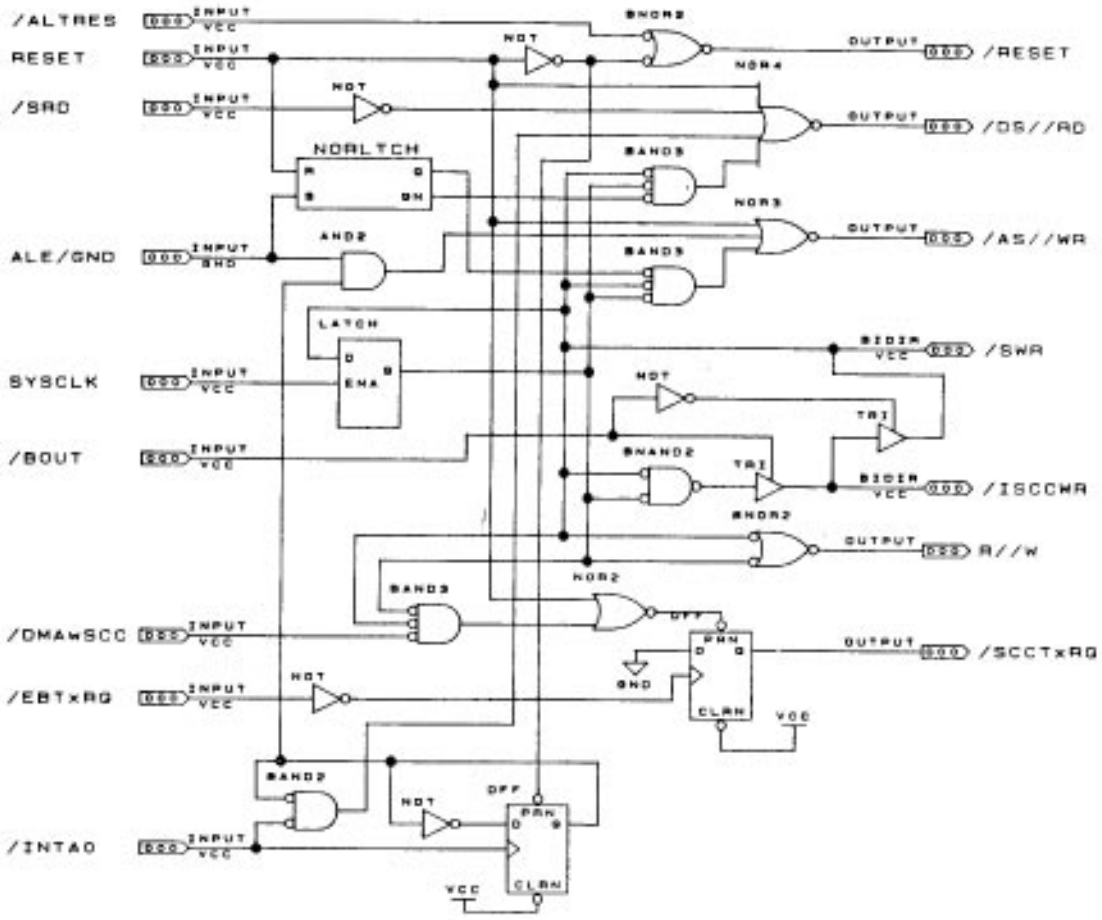



**Figure 2. SCC EPLD for 186 Board**

**DMA/EPLD LOGIC** (Continued)

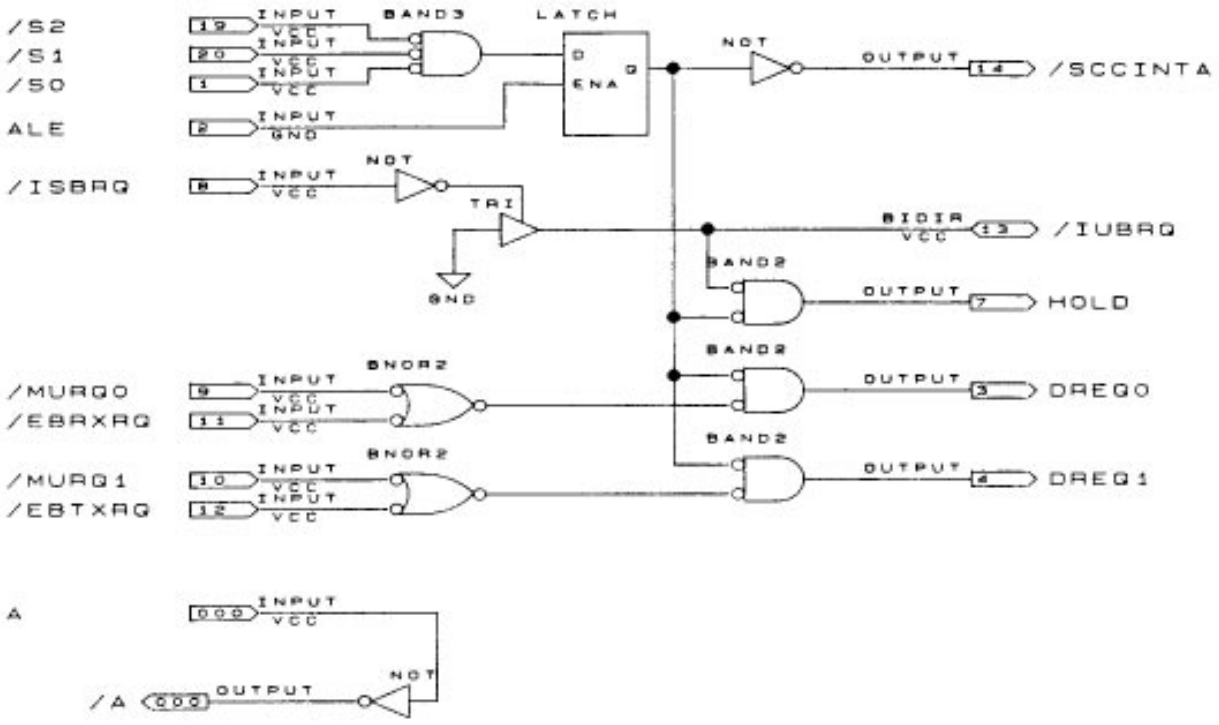



**Figure 3. DMA EPLD for 186 Board**

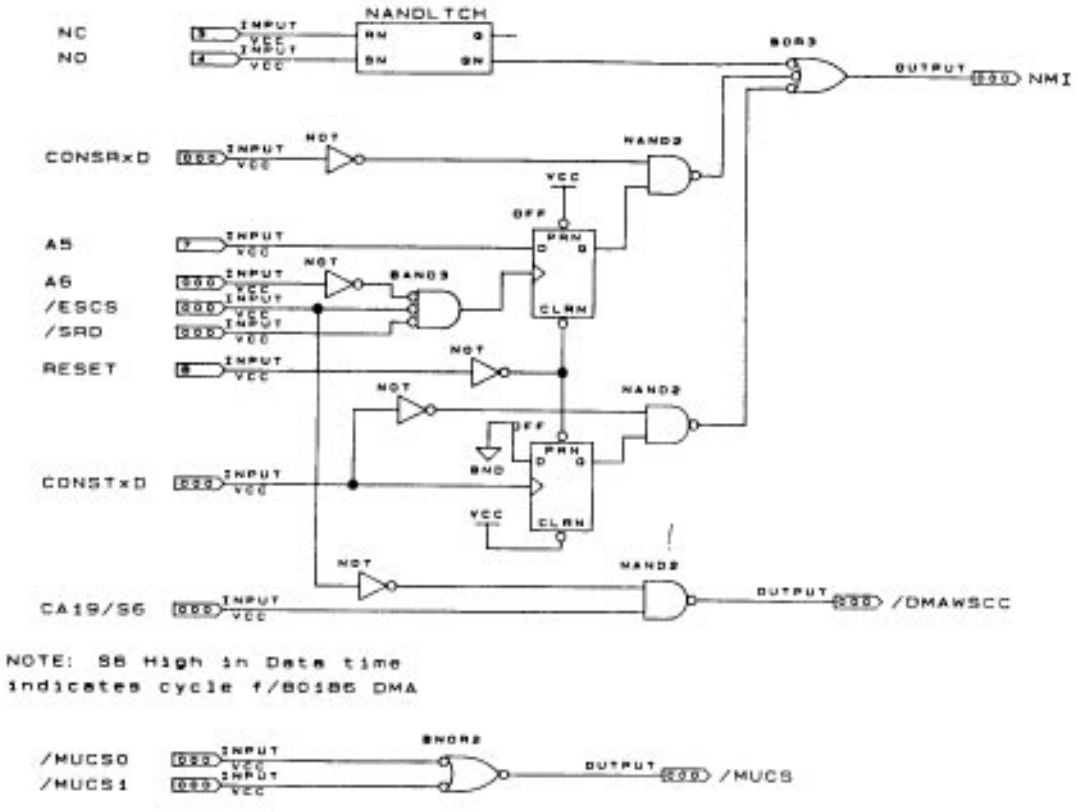



**Figure 4. NMI Field for 186 Board**

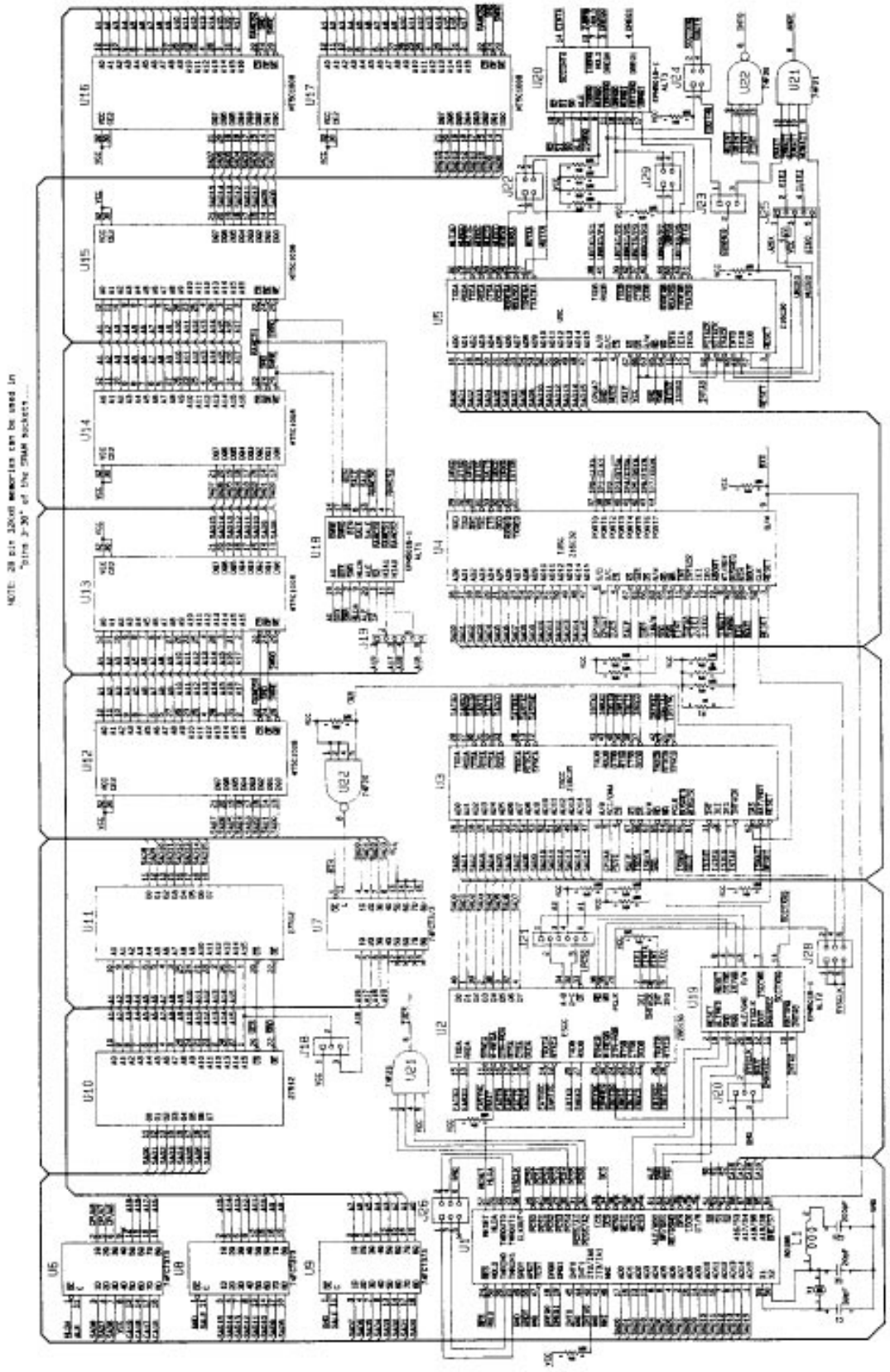


**Figure 5. Schematic of the Evaluation Board**

# SCC IN BINARY SYNCHRONOUS COMMUNICATIONS

## INTRODUCTION

Zilog's Z8030 Z-SCC Serial Communications Controller is one of a family of components that are Z-BUS® compatible with the Z8000™ CPU. Combined with a Z8000 CPU (or other existing 8- or 16-bit CPUs with nonmultiplexed buses when using the Z8530 SCC), the Z-SCC forms an integrated data communications controller that is more cost effective and more compact than systems incorporating UARTs, baud rate generators, and phase-locked loops as separate entities.

The approach examined here implements a communications controller in a Binary Synchronous mode of operation, with a Z8002 CPU acting as controller for the Z-SCC.

One channel of the Z-SCC is used to communicate with the remote station in Half Duplex mode at 9600 bits/second. To test this application, two Z8000 Development Modules are used. Both are loaded with the same software routines for initialization and for transmitting and receiving messages. The main program of one module requests the transmit routine to send a message of the length indicated in the 'COUNT' parameter. The other system receives the incoming data stream, storing the message in its resident memory.

## DATA TRANSFER MODES

The Z-SCC system interface supports the following data transfer modes:

- **Polled Mode.** The CPU periodically polls the Z-SCC status registers to determine the availability of a received character, if a character is needed for transmission, and if any errors have been detected.
- **Interrupt Mode.** The Z-SCC interrupts the CPU when certain previously defined conditions are met.
- **Block/DMA Mode.** Using the Wait/Request (/W//REQ) signal, the Z-SCC introduces extra wait cycles to synchronize data transfer between a CPU or DMA controller and the Z-SCC.

The example given here uses the block mode of data transfer in its transmit and receive routines.

## SYNCHRONOUS MODES

Three variations of character-oriented synchronous communications are supported by the Z-SCC: Mono-sync, Bisync, and External Sync (Figure 1). In Monosync mode, a single sync character is transmitted, which is then compared to an identical sync character in the receiver. When the receiver recognizes this sync character, synchronization is complete; the receiver then transfers subsequent characters into the receiver FIFO in the Z-SCC.

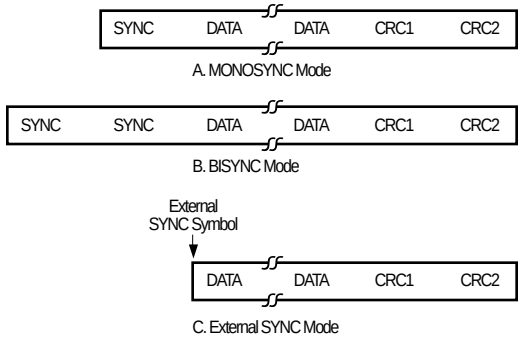



**Figure 1. Synchronous Modes of Communication**

Bisync mode uses a 16-bit or 12-bit sync character in the same way to obtain synchronization. External Sync mode uses an external signal to mark the beginning of the data field; i.e., an external input pin (SYNC) indicates the start of the information field.

In all synchronous modes, two Cyclic Redundancy Check (CRC) bytes can be concatenated to the message to detect data transmission errors. The CRC bytes inserted in the transmitted message are compared to the CRC bytes computed to the receiver. Any differences found are held in the receive error FIFO.

## SYSTEM INTERFACE

The Z8002 Development Module consists of a Z8002 CPU, 16K words of dynamic RAM, 2K words of EPROM monitor, a Z80A SIO providing dual serial ports, a Z80A CTC peripheral device providing four counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire wrap area for prototyping. The block diagram is depicted in Figure 2. Each of the peripherals in the development module is connected in a prioritized daisy-chain configuration. The Z-SCC is included in this configuration by tying its IEI line to the IEO line of another device, thus making it one stop lower in interrupt priority compared to the other device.

Two Z8000 Development Modules containing Z-SCCs are connected as shown in Figure 3 and Figure 4. The Transmit Data pin of one is connected to the Receive Data pin of the other and vice versa. The Z8002 is used as a host CPU for loading the modules' memories with software routines.

The Z8000 CPU can address either of the two bytes contained in 16-bit words. The CPU uses an even address (16 bits) to access the most-significant byte of a word and an odd address for the least-significant byte of a word.

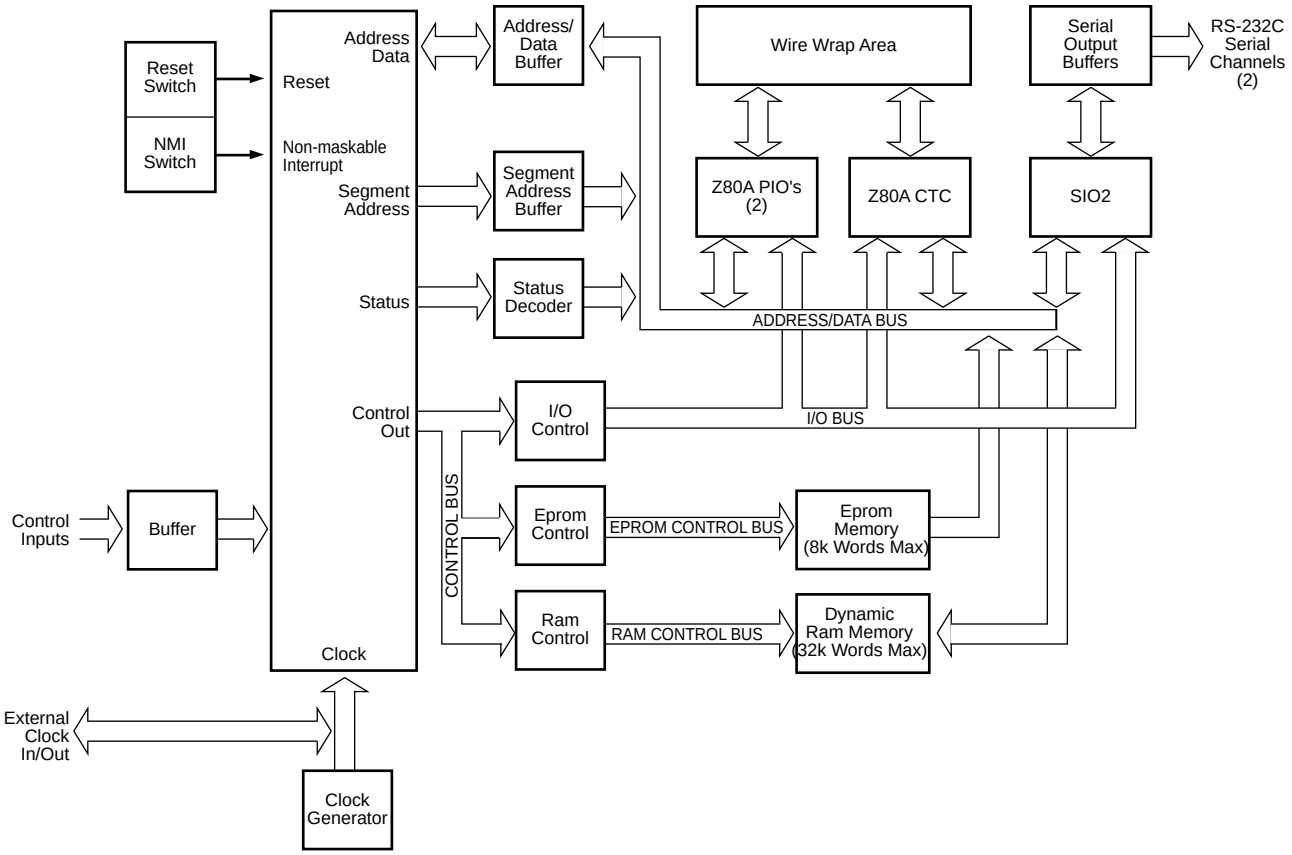



**Figure 2. Block Diagram of Z8000 DM**

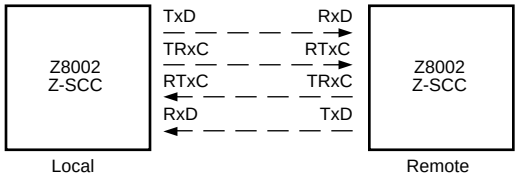



**Figure 3. Block Diagram of Two Z8000 Development Modules**

## SYSTEM INTERFACE (Continued)

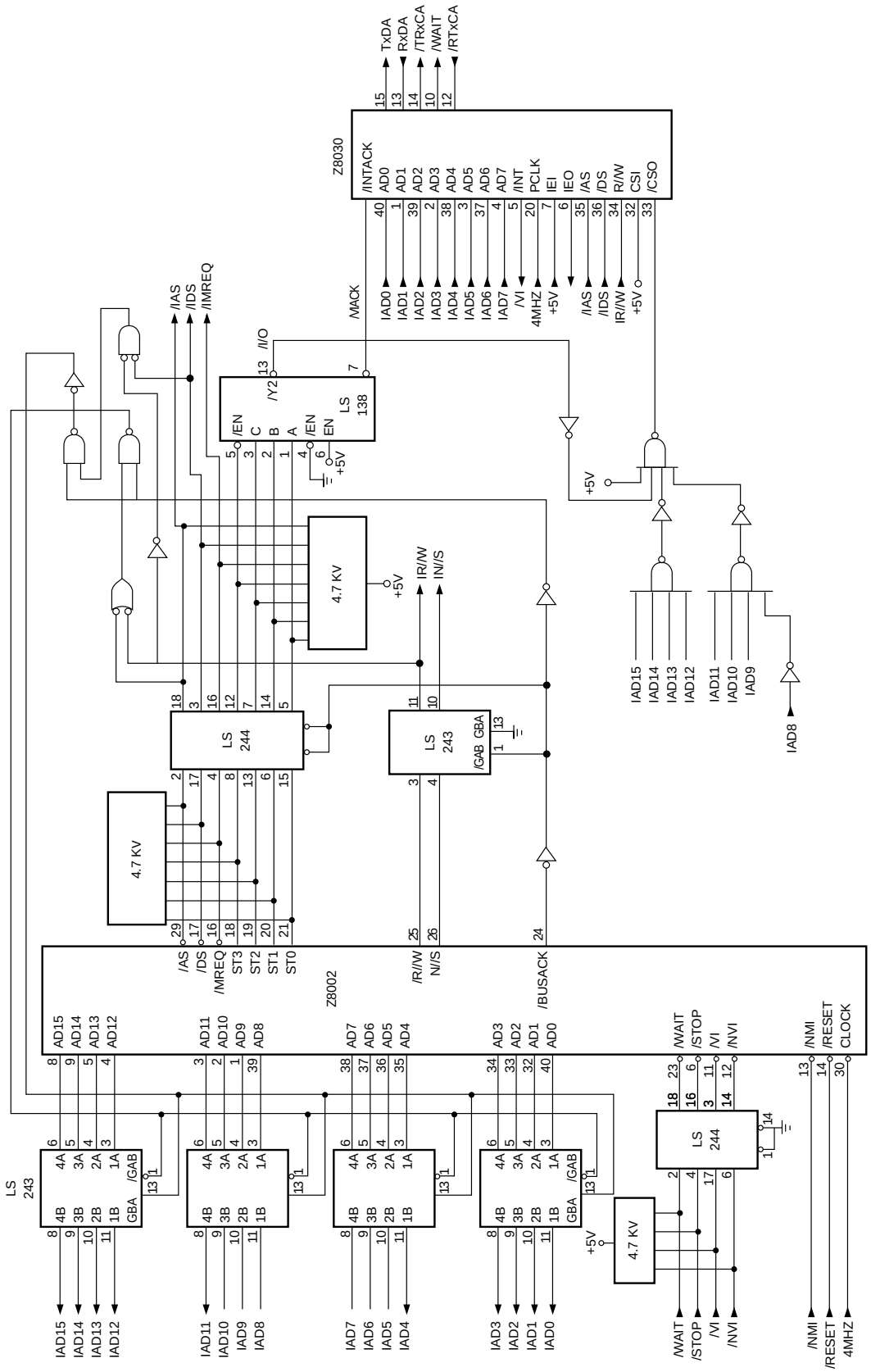



**Figure 4. Z8002 with SCC**

When the Z8002 CPU uses the lower half of the Address/Data bus (AD0-AD7 the least significant byte) for byte read and write transactions during I/O operations, these transactions are performed between the CPU and I/O ports located at odd I/O addresses. Since the Z-SCC is attached to the CPU on the lower half of the A/D bus, its registers must appear to the CPU at odd I/O addresses. To achieve this, the Z-SCC can be programmed to select its internal registers using lines AD5-AD1. This is done either automatically with the Force Hardware Reset command in WR9 or by sending a Select Shift Left Mode command to WR0B in channel B of the Z-SCC. For this application, the Z-SCC registers are located at I/O port address 'FExx'. The Chip Select signal (/CS0) is derived by decoding I/O address 'FE' hex from lines AD15-AD8 of the controller. The Read/Write registers are automatically selected by the Z-SCC when internally decoding lines AD5-AD1 in Shift Left mode. To select the Read/Write registers automatically, the Z-SCC decodes lines AD5-AD1 in Shift Left mode. The register map for the Z-SCC is depicted in Table 1.

## INITIALIZATION

The Z-SCC can be initialized for use in different modes by setting various bits in its Write registers. First, a hardware reset must be performed by setting bits 7 and 6 of WR9 to one; the rest of the bits are disabled by writing a logic zero.

Bisync mode is established by selecting a 16-bit sync character, Sync Mode Enable, and a Xl clock in WR4. A data rate of 9600 baud, NRZ encoding, and a data character length of eight bits are among the other options that are selected in this example (Table 2).

Note that WR9 is accessed twice, first to perform a hardware reset and again at the end of the initialization sequence to enable the interrupts. The programming sequence depicted in Table 2 establishes the necessary parameters for the receiver and the transmitter so that, when enabled, they are ready to perform communication tasks. To avoid internal race and false interrupt conditions, it is important to initialize the registers in the sequence depicted in this application note.

**Table 1. Register Map**

| Address (hex) | Write Register | Read Register |
|---|---|---|
| FE01 | WR0B | RR0B |
| FE03 | WR1B | RR1B |
| FE05 | WR2 | RR2B |
| FE07 | WR3B | RR3B |
| FE09 | WR4B | |
| FE0B | WR5B | |
| FE0D | WR6B | |
| FE0F | WR7B | |
| FE11 | B DATA | B DATA |
| FE13 | WR9 | |
| FE15 | WR10B | RR10B |
| FE17 | WR11B | |
| FE19 | WR12B | RR12B |
| FE1B | WR13B | RR13B |
| FE1D | WR14B | |
| FE1F | WR15B | RR15B |
| FE21 | WR0A | RR0A |
| FE23 | WR1A | RR1A |
| FE25 | WR2 | RR2A |
| FE27 | WR3A | RR3A |
| FE29 | WR4A | |
| FE2B | WR5A | |
| FE2D | WR6A | |
| FE2F | WR7A | |
| FE31 | A DATA | A DATA |
| FE33 | WR9 | |
| FE35 | WR10A | RR10A |
| FE37 | WR11A | |
| FE39 | WR12A | RR12A |
| FE3B | WR13A | RR13A |
| FE3D | WR14A | |
| FE3F | WR15A | RR15A |

## INITIALIZATION (Continued)

The Z8002 CPU must be operated in System mode in order to execute privileged I/O instructions, so the Flag Control Word (FCW) should be loaded with System/Normal (S//N), and the Vectored Interrupt Enable (VIE) bits set. The Program Status Area Pointer (PSAP) is loaded with address %4400 using the Load Control instruction (LDCTL). If the Z8000 Development Module is intended to be used, the PSAP need not be loaded by the programmer as the development modules monitor loads it automatically after the NMI button is pressed.

**Table 2. Programming Sequence for Initialization**

| Register | Value (hex) | Effect |
|---|---|---|
| WR9 | C0 | Hardware reset |
| WR4 | 10 | x1 clock, 16-bit sync, sync mode enable |
| WR10 | 0 | NRZ, CRC preset to zero |
| WR6 | AB | Any sync character “AB” |
| WR7 | CD | Any sync character “CD” |
| WR2 | 20 | Interrupt vector “20” |
| WR11 | 16 | Tx clock from BRG output, TRxC pin = BRG out |
| WR12 | CE | Lower byte of time constant = “CE” for 9600 baud |
| WR13 | 0 | Upper byte = 0 |
| WR14 | 03 | BRG source bit = 1 for PCLK as input, BRG enable |
| WR15 | 00 | External interrupt disable |
| WR5 | 64 | Tx 8 bits/character, CRC-16 |
| WR3 | C1 | Rx8 bits/character, Rx enable (Automatic Hunt mode) |
| WR1 | 08 | RxInt on 1st char & sp. cond., ext. int. disable) |
| WR9 | 09 | MIE, VIS, Status Low |

Since VIS and Status Low are selected in WR9, the vectors listed in Table 3 will be returned during the Interrupt Acknowledge cycle. Of the four interrupts listed, only two, Ch A Receive Character Available and Ch A Special Receive Condition, are used in the example given here.

**Table 3. Interrupt Vectors**

| Vector (hex) | PS Address* (hex) | Interrupt |
|---|---|---|
| 28 | 446E | Ch A Transmit Buffer Empty |
| 2A | 4472 | Ch A External Status Change |
| 2C | 4476 | Ch A Receive Char. Available |
| 2E | 447A | Ch A Special Receive Condition |

* “PS Address” refers to the location in the Program Status Area where the service routine address is stored for that particular interrupt, assuming that PSAP has been set to 4400 hex.

## TRANSMIT OPERATION

To transmit a block of data, the main program calls up the transmit data routine. With this routine, each message block to be transmitted is stored in memory, beginning with location 'TBUF'. The number of characters contained in each block is determined by the value assigned to the 'COUNT' parameter in the main module.

To prepare for transmission, the routine enables the transmitter and selects the Wait On Transmit function; it then enables the wait function. The Wait On Transmit function indicates to the CPU whether or not the Z-SCC is ready to accept data from the CPU. If the CPU attempts to send data to the Z-SCC when the transmit buffer is full, the Z-SCC asserts its Wait line and keeps it Low until the buffer is empty. In response, the CPU extends its I/O cycles until the Wait line goes inactive, indicating that the Z-SCC is ready to receive data.

The CRC generator is reset and the Transmit CRC bit is enabled before the first character is sent, thus including all the characters sent to the Z-SCC in the CRC calculation, until the Transmit CRC bit is disabled. CRC generation can be disabled for a particular character by resetting the TxCRC bit within the transmit routine. In this application, however, the Transmit CRC bit is not disabled, so that all characters sent to the Z-SCC are included in the CRC calculation.

The Z-SCC's transmit underrun/EOM latch must be reset sometime after the first character is transmitted by writing a Reset Tx Underrun/EOM command to WR0. When this latch is reset, the Z-SCC automatically appends the CRC characters to the end of the message in the case of an underrun condition.

Finally, a five-character delay is introduced at the end of the transmission, which allows the Z-SCC sufficient time to transmit the last data byte, two CRC characters, and two sync characters before disabling the transmitter.

## RECEIVE OPERATION

Once the Z-SCC is initialized, it can be prepared to receive data. First, the receiver is enabled, placing the Z-SCC in Hunt mode and thus setting the Sync/Hunt bit in status register RR0 to 1. In Hunt mode, the receiver is idle except that it searches the incoming data stream for a sync character match. When a match is discovered between the incoming data stream and the sync characters stored in WR6 and WR7, the receiver exits the Hunt mode, resetting the Sync/Hunt bit in status register RR0 and establishing the Receive Interrupt On First Character mode. Upon detection of the receive interrupt, the CPU generates an Interrupt Acknowledge cycle. The Z-SCC sends to the CPU vector %2C, which points to the location in the Program Status Area from which the receive interrupt service routine is accessed.

The receive data routine is called from within the receive interrupt service routine. While expecting a block of data, the Wait On Receive function is enabled. Receive data buffer RR8 is read, and the characters are stored in memory locations starting at RBUF. The Start of Text (%02) character is discarded. After the End of Transmission character (%04) is received, the two CRC bytes are read. The result of the CRC check becomes valid two characters later, at which time, RR1 is read and the CRC error bit is checked. If the bit is zero, the message received can be assumed correct; if the bit is 1, an error in the transmission is indicated.

Before leaving the interrupt service routine, Reset Highest IUS (Interrupt Under Service), Enable Interrupt on Next Receive Character, and Enter Hunt Mode commands are issued to the Z-SCC.

If a receive overrun error is made, a special condition interrupt occurs. The Z-SCC presents the vector %2E to the CPU, and the service routine located at address %447A is executed. The Special Receive Condition register RR1 is read to determine which error occurred. Appropriate action to correct the error should be taken by the user at this point. Error Reset and Reset Highest IUS commands are given to the Z-SCC before returning to the main program so that the other lower priority interrupts can occur.

## SOFTWARE

Software routines are presented in the following pages. These routines can be modified to include various versions of Bisync protocol, such as Transparent and Nontransparent modes. Encoding methods other than NRZ (e.g., NRZI, FM0, FM1) can also be used by modifying WR10.

# APPENDIX

## SOFTWARE ROUTINES

**plzasm 1.3**

```
LOC  OBJ CODE  STMT SOURCE    STATEMENT
                1             BISYNC MODULE
                    $LISTON   $TTY
                    CONSTANT
                    WR0A      :=    %FE21       !BASE ADDRESS FOR WR0 CHANNEL A!
                    RR0A      :=    %FE21       !BASE ADDRESS FOR RR0 CHANNEL A!
                    RBUF      :=    %5400       !BUFFER AREA FOR RECEIVE CHARACTER!
                    PSAREA    :=    %4400       !START ADDRESS FOR PROGRAM STAT AREAS!
                    COUNT     :=    12          !NO. OF CHAR FOR TRANSMIT ROUTINE!
0000                GLOBAL MAIN PROCEDURE
                    ENTRY
0000 7601                     LDA   R1, PSAREA
0002 4400
0004 7D1D                     LDCTL PSAPOFF,R1  !LOAD PSAP
0006 2100                     LD    RO,#%5000
0008 5000
000A 3310                     LD    RI(#%IC),R0 !FCW VALUE(%5000) AT %441C FOR VECTORED!
000C 001C                                       !INTERRUPTS!
000E 7600                     LDA   R0,REC
0010 00F4'
0012 3310                     LD    RI(#%76),R0 !EXT. STATUS SERVICE ADDR. AT %4476 IN!
0014 0076                                       !PSA!
0016 7600                     LDA   R0, SPCOND
0018 011E'
00IA 3310                     LD    R1(#%7A),R0 !SP.COND.SERVICE ADDR AT %447A IN PSA!
001C 007A
001E 5F00                     CALL  INIT
0020 0034'
0022 5F00                     CALL  TRANSMIT
0024 00A6'
0026 E8FF                     JR    $
0028 02         TBUF:         BVAL  %02         !START OF TEXT!
0029 31                       BVAL  '1'         !BVAL MEANS BYTE VALUE. MESSAGE CHAR.!
002A 32                       BVAL  '2'
002B 33                       BVAL  '3'
002C 34                       BVAL  '4'
002D 35                       BVAL  '5'
002E 36                       BVAL  '6'
002F 37                       BVAL  '7'
0030 38                       BVAL  '8'
0031 39                       BVAL  '9'
0032 30                       BVAL  '0'
0033 31                       BVAL  '1'
0034                          END   MAIN
```

## INITIALIZATION ROUTINE FOR Z-SCC

| 0034 | | GLOBAL ENTRY | INIT | PROCEDURE | |
|---|---|---|---|---|---|
| 0634 | 2100 | | LD | R0, #15 | !NO.OF PORTS TO WRITE TO! |
| 0036 | 000F | | | | |
| 0038 | 7602 | | LDA | R2, SCCTAB | !ADDRESS OF DATA FOR PORTS! |
| 003A | 004E' | | | | |
| 003C | 2101 | ALOOP: | LD | R1, #WR0A | |
| 003E | FE21 | | | | |
| 0040 | 0029 | | ADDB | RL1, @R2 | |
| 0042 | A920 | | INC | R2 | |
| 0044 | 3A22 | | OUTIB | @Rl, @R2,R0 | !POINT TO WR0A,WR1A ETC THRO LOOP! |
| 0046 | 0018 | | | | |
| 0048 | 8D04 | | TEST | R0 | !END OF LOOP?! |
| 004A | EEF8 | | JR | NZ, ALOOP | !NO, KEEP LOOPING! |
| 004C | 9E08 | | RET | | |
| 004E | 12 | SCCTAB: | BVAL | 2*9 | |
| 004F | CO | | BVAL | %C0 | !WR9=HARDWARE RESET! |
| 0050 | 08 | | BVAL | 2*4 | |
| 0051 | 10 | | BVAL | %10 | !WR4=X1 CLK, 16 BIT SYNC MODE! |
| 0052 | 14 | | BVAL | 2*10 | |
| 0053 | 00 | | BVAL | 0 | !WRIO=CRC PRESET ZERO, NRZ,16 BIT SYNC! |
| 0054 | 0C | | BVAL | 2*6 | |
| 0055 | AB | | BVAL | %AB | !WR6=ANY SYNC CHAR %AB! |
| 0056 | 0E | | BVAL | 2*7 | |
| 0057 | CD | | BVAL | %CD | !WR7=ANY SYNC CHARR %CD! |
| 0058 | 04 | | BVAL | 2*2 | |
| 0059 | 20 | | BVAL | %20 | !WR2=NT VECTOR %20! |
| 005A | 16 | | BVAL | 2*11 | |
| 005B | 16 | | BVAL | %16 | !WR11=TxCLOCK & TRxC OUT=BRG OUT! |
| 005C | 18 | | BVAL | 2*12 | |
| 005D | CE | | BVAL | %CE | !WR12= LOWER TC=%CE! |
| 005E | IA | | BVAL. | 2*13 | |
| 005F | 00 | | BVAL | 0 | !WR13= UPPER TC=01 |
| 0060 | 1C | | BVAL | 2*14 | |
| 0061 | 03 | | BVAL | %03 | !WRI4=BRG ON, ITS SRC=PCLK! |
| 0062 | 1E | | BVAL | 2*15 | |
| 0063 | 00 | | BVAL | %00 | !WRI5=NO EXT INT EN.! |
| 0064 | 0A | | BVAL | 2*5 | |
| 0065 | 64 | | BVAL | %64 | !WR5= TX 8 BITS/CHAR, CRC-16! |
| 0066 | 06 | | BVAL | 2*3 | |
| 0067 | Cl | | BVAL | &CI | IWR3=RX 8 BITS/CHAR, REC ENABLE! |
| 0068 | 02 | | BVAL | 2*1 | |
| 0069 | 08 | | BVAL | %C1 | !WR1=RxINT ON 1ST OR SP COND!<br>!EXT INT DISABLE! |
| 006A | 12 | | BVAL | 2*9 | |
| 006B | 09 | | BVAL | %09 | !WR9=MIE, VIS, STATUS LOW! |
| 006C | | END INIT | | | |

## RECEIVE ROUTINE

RECEIVE A BLOCK OF MESSAGE
THE LAST CHARACTER SHOULD BE EOT (%04)

```
006C            GLOBAL
006C            ENTRY   RECEIVE PROCEDURE
006C C828               LDB     RL0,#428        !WAIT ON RECV.!
006C 3A86               OUTB    WR0A+2,RL0
0070 FE23
0072 6000               LDB     RL0,%A8
0074 00AB
0076 3A86               OUTB    WR0A+2,RL0      !ENABLE WAIT 1ST CHAR,SP.COND. INT!
0078 FE23
007A 2101               LD      Rl,#RR0A+16
007C FE31
007E 3Cl8               INB     RL0,@R1         !READ STX CHARACTER!
0080 C8C9               LDB     RL0,#%C9
0082 3AB6               OUTB    WR0A+6,RL0      !Rx CRC ENABLE!
0084 FE27
0086 2103               LD      R3,#RBUF
0088 5400
008A 3C18       READ:   INB     RL0,@R1         !READ MESSAGE!
008C 2E38               LDB     @R3,RL0         !STORE CHARACTER IN RBUF!
008E AB30               DEC     R3,#l
0090 0A08               CPB     RL0,#%04        !IS IT END OF TRANSMISSION ?!
0092 0404
0094 EEFA               JR      NZ,READ
0096 3C18               INB     RL0,@R1         !READ PAD1!
0098 3C18               INB     RL0,@R1         !READ PAD2!
009A 3A84               INB     RL0,RR0A+2      !READ CRC STATUS!
009C FE23
                ! PROCESS CRC ERROR IF ANY, AND GIVE ERROR RESET COMMAND IN WR0A!
009E C800               LDB     RL0,#0
00A0 3A86               OUTB    WR0A+6,RL0      !DISABLE RECEIVER!
00A2 FE27
00A4 9E08               RET
00A6            END RECEIVE
```

## TRANSMIT ROUTINE

SEND A BLOCK OF DATA CHARACTERS
THE BLOCK STARTS AT LOCATION TBUP

| OA6 | | GLOBAL ENTRY | TRANSMIT | PROCEDURE | |
|---|---|---|---|---|---|
| 00A6 | 2102 | | LD | R2, #TBUF | !PTR TO START OF BUFFER! |
| 00AB | 0028' | | | | |
| 00AA | C86C | | LDB | RL0, #%6C | |
| 00AC | 3AB6 | | OUTB | WR0A+10, RL0 | !ENABLE TRANSMITTER! |
| 00AE | FE2B | | | | |
| 00B0 | C800 | | LDB | RL0, #%00 | !WAIT ON TRANSMIT! |
| 00B2 | 3A86 | | OUTB | WR0A+2 , RL0 | |
| 00B4 | FE23 | | | | |
| 00B6 | C888 | | LDB | RL0, #%88 | |
| 00B8 | 3AB6 | | OUTB | WR0A+2, RL0 | !WAIT ENABLE, INT ON 1ST & SP COND! |
| 00BA | FE23 | | | | |
| 00BC | C880 | | LDB | RL0, #%80 | |
| 00BE | 3A86 | | OUTB | WR0A, RL0 | !RESET TxCRC GENERATOR! |
| 00C0 | FE21 | | | | |
| 00C2 | 2101 | | LD | R1, #WR0A+16 | !WR8A SELECTED! |
| 00C4 | FE31 | | | | |
| 00C6 | C86D | | LDB | RL0, #%6D | |
| 00C8 | 3A86 | | OUTB | WR0A+10, RL0 | !Tx CRC ENABLE! |
| 00CA | FE2B | | | | |
| 00CC | 2100 | | LD | R0, #1 | |
| 00CE | 0001 | | | | |
| 00D0 | 3A22 | | OTIRB | @Rl, @R2,R0 | !SEND START OF TEXT! |
| 00D2 | 0010 | | | | |
| 00D4 | C8C0 | | LDB | RL0, #%C0 | |
| 00D6 | 3AB6 | | OUTB | WR0A, RL0 | !RESET TxUND/EOM LATCH! |
| 00D8 | FE21 | | | | |
| 00DA | 2100 | | LD | R0, #COUNT-1 | |
| 00DC | 000B | | | | |
| 00DE | 3A22 | | OTIRB | @Rl, @R2, R0 | !SEND MESSAGE! |
| 00E0 | 0010 | | | | |
| 00E2 | C804 | | LDB | RL0, #%04 | |
| 00E4 | 3EI8 | | OUTB | @R1, RL0 | !SEND END OF TRANSMISSION CHARACTER! |
| 00E6 | 2100 | | LD | R0, #1670 | !CREATE DELAY BEFORE DISABLING! |
| 00E8 | 0686 | | | | |
| 00EA | F081 | DEL: | DJNZ | R0, DEL | |
| 00EC | C800 | | LDB | RL0, #0 | |
| 00EE | 3AB6 | | OUTB | WR0A+10, RL0 | !DISABLE TRANSMITTER! |
| 00F0 | FE2B | | | | |
| 00F2 | 9E0B | | RET | | |
| 00F4 | | END TRANSMIT | | | |

## RECEIVE INT. SERVICE ROUTINE

| 00F4 | | GLOBAL ENTRY | REC | PROCEDURE | |
|---|---|---|---|---|---|
| 00F4 | 93F0 | | PUSH | @RI5, R0 | |
| 00F6 | 3A84 | | INB | RL0, RR0A | !READ STATUS FROM RR0A! |
| 00F8 | FE21 | | | | |
| 00FA | A684 | | BITB | RL0, #4 | !TEST IF SYNC HUNT RESET! |
| 00FC | EE02 | | JR | NZ, RESET | !YES CALL RECEIVE ROUTINE! |
| 00FE | 5F00 | | CALL | RECEIVE | |
| 0100 | 006C’ | | | | |
| 0102 | C808 | RESET: | LDB | RL0, #%08 | |
| 0104 | 3A86 | | OUTB | WR0A+2, RL0 | !WAIT DISABLE! |
| 0106 | FE23 | | | | |
| 0108 | C8D1 | | LDB | RL0, #%D1 | |
| 010A | 3A86 | | OUTB | WR0A+6, RL0 | !ENTER HUNT MODE! |
| 010C | FE27 | | | | |
| 010E | C820 | | LDB | RL0, #%20 | |
| 0110 | 3A86 | | OUTB | WR0A, RL0 | !ENABLE INT ON NEXT CHAR! |
| 0112 | FE21 | | | | |
| 0114 | C838 | | LDB | RL0, #%38 | |
| 0116 | 3A86 | | OUTB | WR0A, RL0 | !RESET HIGHEST IUS! |
| 0118 | FE21 | | | | |
| 011A | 97F0 | | POP | R0, @RI5 | |
| 011C | 7B00 | | IRET | | |
| 0IIE | | END REC | | | |

## SPECIAL CONDITION INTERRUPT SERVICE ROUTINE

```
                        GLOBAL
011E                    ENTRY   SPCOND  PROCEDURE
011E    93F0                    PUSH    @Rl5, R0
0120    3A84                    INB     RL0, RR0A+2     !READ ERRORS!
0122    FE23
                                !PROCESS ERRORS!
0124    C830                    LDB     RL0, #%30
0126    3A8B6                   OUTB    WR0A, RL0       !ERROR RESET!
0128    FE21
012A    C808                    LDB     RL0, #%08
012C    3A86                    OUTB    WR0A+2, RL0     !WAIT DISABLE, RxINT ON 1ST OR SP COND.!
012E    FE23
0130    C0D1                    LDB     RL0, #%D1
0132    3A86                    OUTB    WR0A+6, RL0     !HUNT MODE, REC. ENABLE!
0134    FE27
0136    C838                    LDB     RL0, #%38
0138    3A86                    OUTB    WR0A, RL0       !RESET HIGHEST IUS!
013A    FE21
013C    97F0                    POP     R0, @Rl5
013E    7B00                    IRET

0140                    END SPCOND

                        END BISYNC
```

o errors
Assembly complete

# SERIAL COMMUNICATION CONTROLLER (SCC™): SDLC MODE OF OPERATION

*Understanding the transactions which occur within a Serial Communication Controller operating in the SDLC mode simplifies working in this complex area.*

## INTRODUCTION

Zilog's SCC (Serial Communication Controller) is a popular USART (Universal Synchronous/Asynchronous Receiver/Transmitter) device, used for a wide range of applications. For instance, Macintosh systems use the SCC as a standard communication controller device. There are several different types of devices in the SCC family. The family consists of the Z8530 NMOS SCC, the Z85C30 CMOS SCC, the Z85230 ESCC (Enhanced SCC), Z85233 EMSCC (Mono Enhanced SCC), and Superintegration devices such as the Z181 ZIO™ and Z182 ZIP™.

Since the SCC may be used in many different ways, it may not be easy to understand all the transactions involved between the CPU and the SCC. In particular, the SDLC mode of operation is highly complicated, and many transactions are involved to make it work properly. This application note describes the sequence of events which occurs in the SDLC mode of operation.

The following sequences of events are covered:

- SDLC Transmission
- SDLC receive
  - With Receive Interrupts on all received characters or Special Conditions
  - With Receive Interrupts on First Character or Special Condition
  - With Receive Interrupts on Special Conditions only
- Receiving Back-to-back Frame under DMA control
- SDLC Loop mode

Each section explains the transmit/receive process for packets with the following characteristics:

- Initial state is mark idle
- Address field has 81H
- Control field has 42H
- Two bytes of I-field, 42H and 0FFH
- After the closing flag, mark idling

**Note:**
This application note describes the SCC, but not the ESCC. The ESCC, since it incorporates enhancements like deeper FIFOs and SDLC mode supporting logic, handles the packets much more simply than the SCC. Refer to the section on CMOS SCC and ESCC of this appnote for more general information on the ESCC.

## SDLC TRANSMIT

Figure 1 shows the time chart for the transmitting SDLC packet under interrupt control. When transmit is engaged, data is shifted out of the transmitter on the falling edge of the transmit clock. Transitions on the diagram are shown coincident with TxCLK fall — in actual practice, there are some associated delay times (which are specified in the data sheet).

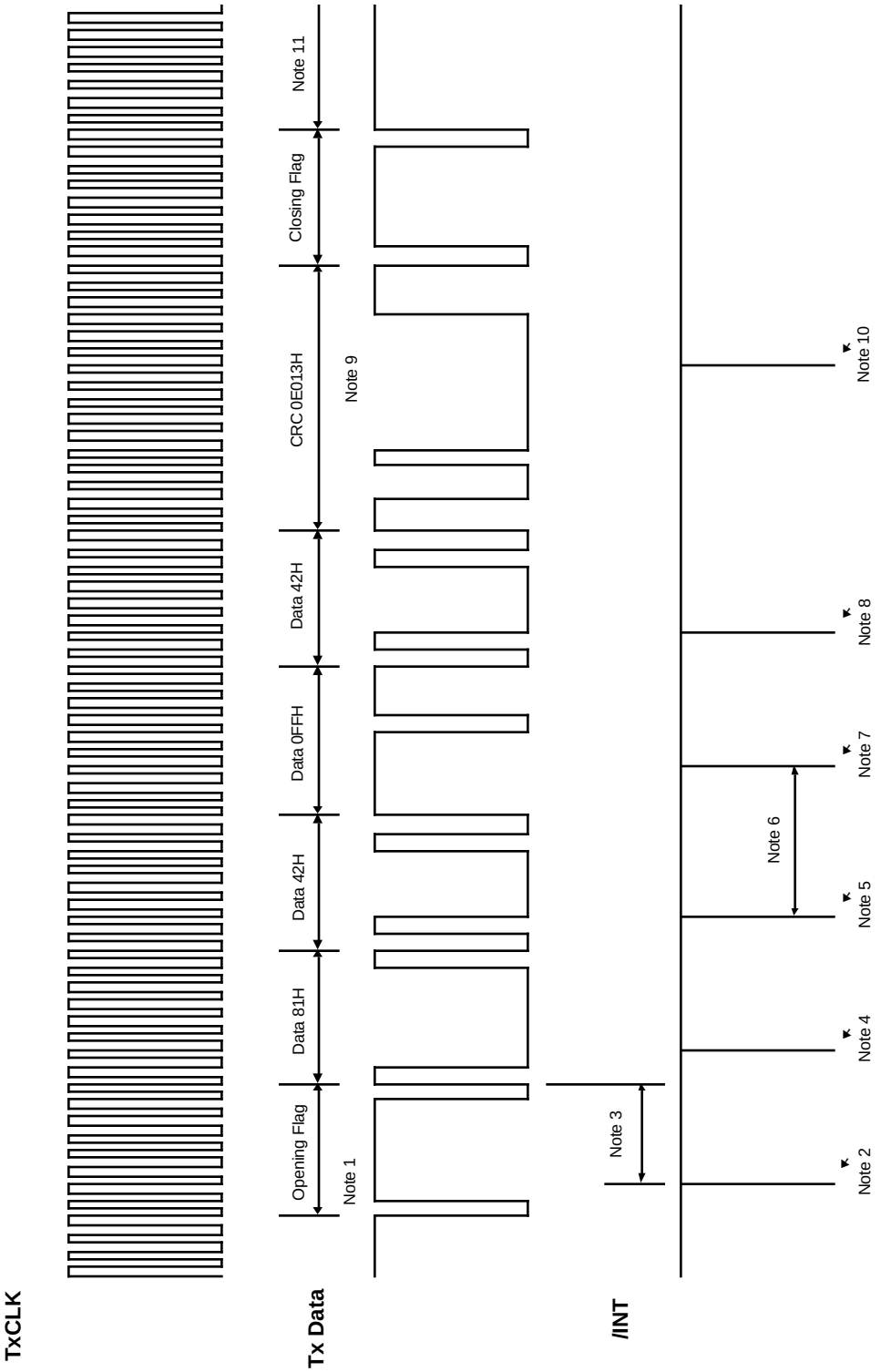



**Figure 1. Typical SDLC Trsnsmission Sequence**

Notes on Figure 1:

1. The SCC has two possible idle states, Mark idle (contiguous logic 1) or Flag idle (repeating flag pattern 7EH). In this figure, the SCC has to be switched to flag idle in order to send the opening flag of the frame. Care must be taken not to put the first data byte (in this case, address 81H) into the Transmit Buffer too soon after the switchover from Mark idle to Flag idle has been made; otherwise, the data may be loaded into the Transmit Shift Register before the flag is loaded. To ensure that this cannot happen, a delay must be executed before the first data byte is put into the buffer. The delay time is dependent on the data rate and a safe minimum duration is 8 bit-times.

2. Transmit Buffer Empty Interrupt for 81H. At this point the data has just been transferred to the Transmit Shift Register and data 42H is written to the Transmit Buffer.

3. The time between the first data byte being transferred to the Shift Register and the first bit appearing at the TxD pin is always six bit-times.

4. Transmit Buffer Empty Interrupt for data 42H. Data 0FFH is written to the Transmit Buffer at this point.

5. Transmit Buffer Empty Interrupt for data 0FFH. Data 42H is written to the Transmit Buffer at this point.

6. The time between interrupts depends on the data character length and the number of zero insertions in the character. For 8 bits/character it can vary between 8 and 10 bit-times. The particular instance shown corresponds to the single zero insertion when the byte 0FFH is transmitted.

7. Transmit Buffer Empty Interrupt for data 42H. Since this is the last byte to be transmitted, the Reset Transmit Interrupt Pending command is issued instead of writing another byte to the Transmit Buffer.

8. Transmitter Underrun/EOM Interrupt. This occurs when both the Transmit Shift Register and the Transmit Buffer are empty. It is an External/Status interrupt. The data sent when this occurs is summarized in the table below:

| Abort/Flag on Underrun bit | Tx Underrun/EOM Latch State when Underrun occurs | Data Sent |
|---|---|---|
| 0 | Reset | CRC and Flag |
| 1 | Reset | Abort and Flag |
| 0 | Set | Flags |
| 1 | Set | Flags |

9. The transmitted CRC is 16 bits long provided that there are no zero insertions. In theory it could be as long as 19 bits.

10. The last interrupt generated occurs after the CRC is shifted out of the transmitter and a flag is loaded to be sent. It is a Transmit Buffer Empty Interrupt. If another frame is to be transmitted, the first character of the next frame can be loaded. The two frames will then be separated by a single flag (Back-to-back frame).

11. If the SCC is set up for mark on idle and a new character is not loaded when the last interrupt occurs, only a single flag is sent.

## SDLC RECEIVE

There are several different ways to receive a SDLC packet on the SCC; by polling, by Interrupts and by DMA. The SCC has the following four Receive Interrupt Modes:

- Disabled. This should be used in the Polling mode.
- Interrupts on all received characters or Special Conditions. This mode should be used for normal interrupt-driven operation.
- Interrupts on First Character or Special Condition. This mode is intended for received data transfer by the DMA, and enables the DMA when the interrupt is received by the First Character of the packet.
- Interrupt on Special Condition only. This mode allows the DMA to free-run and keep transferring data to the buffer. This is an ideal mode for the CMOS SCC as well as the ESCC with Status FIFO enabled, because the Status FIFO can give byte count and error status without interrupting data transfer operations.

Each of the four cases is covered in this application note except Receive Interrupt disabled. For polling, the basic operation is identical to that used for “interrupt on all characters or Special Condition” mode. Instead of waiting for an interrupt, polling Reads Registers to determine if service is needed or not.

On the SCC, data is sampled by the receiver on the rising edge of the receive clock. Set-up and hold times for RxD with respect to RxC are specified in the product specifications.

In general, receiver status changes are triggered by RxC. In the following Figures, they are shown as being coincident with this edge — in actual practice, there are some associated delay times (which are specified in the data sheet).

## RECEIVE INTERRUPTS ON ALL RECEIVE CHARACTERS OR SPECIAL CONDITIONS

SCC is placed in this mode by programming Bit D4-3 of WR1 to 10. Once programmed in this mode, the SCC generates interrupts whenever character(s) are in the receive buffer or when Special Conditions occur. This mode is the most common operational mode.

**Notes on Figure 2:**

1. The receiver is usually in hunt mode when waiting for a frame. When the opening flag is received, an External/Status Interrupt is generated, indicating the change from hunt mode to sync mode.
2. The /SYNC output follows the state of the sync register comparison output. The comparison is done on a bit by bit basis, so the /SYNC pin is only active for one bit-time. /SYNC goes active one bit-time after the last bit of the sync character is sampled at the RxD pin.
3. A Receive Character Available Interrupt is generated 11 bit-times 8 bits for the shifter and a 3-bit delay) after the last bit of the character is sampled at the RxD pin. The status bits corresponding to that character must be read before the data character is read from the Receive Buffer. This interrupt is for data 81H.
4. Receive Character Available Interrupt for data 42H.
5. Receive Character Available Interrupt for data 0FFH.
6. Receive Character Available Interrupt for data 42H.
7. Receive Character Available Interrupt for the first CRC byte. The SCC treats the CRC as data, since the SCC does not yet distinguish a difference between CRC and data!
8. The closing flag is recognized two bit-times before the second CRC byte is completely assembled in the Receive Shift Register. As soon as it is recognized, a Special Condition interrupt is generated. The EOF bit is set at this point and the CRC error bit can be checked. The six least significant bits of the second CRC byte are present at the top of the first CRC byte. The status information must be read before the second CRC byte is read from the buffer. The CRC bytes should be discarded. The CRC checker is automatically reset for the next frame.
9. External/Status interrupt for the Sync/Hunt change. This occurs when the SCC recognizes an Abort (Marking line) and forces the receiver into hunt mode. The SCC can be programmed so that the Abort itself generates an interrupt if required. If flag idle was set, this interrupt will not occur.

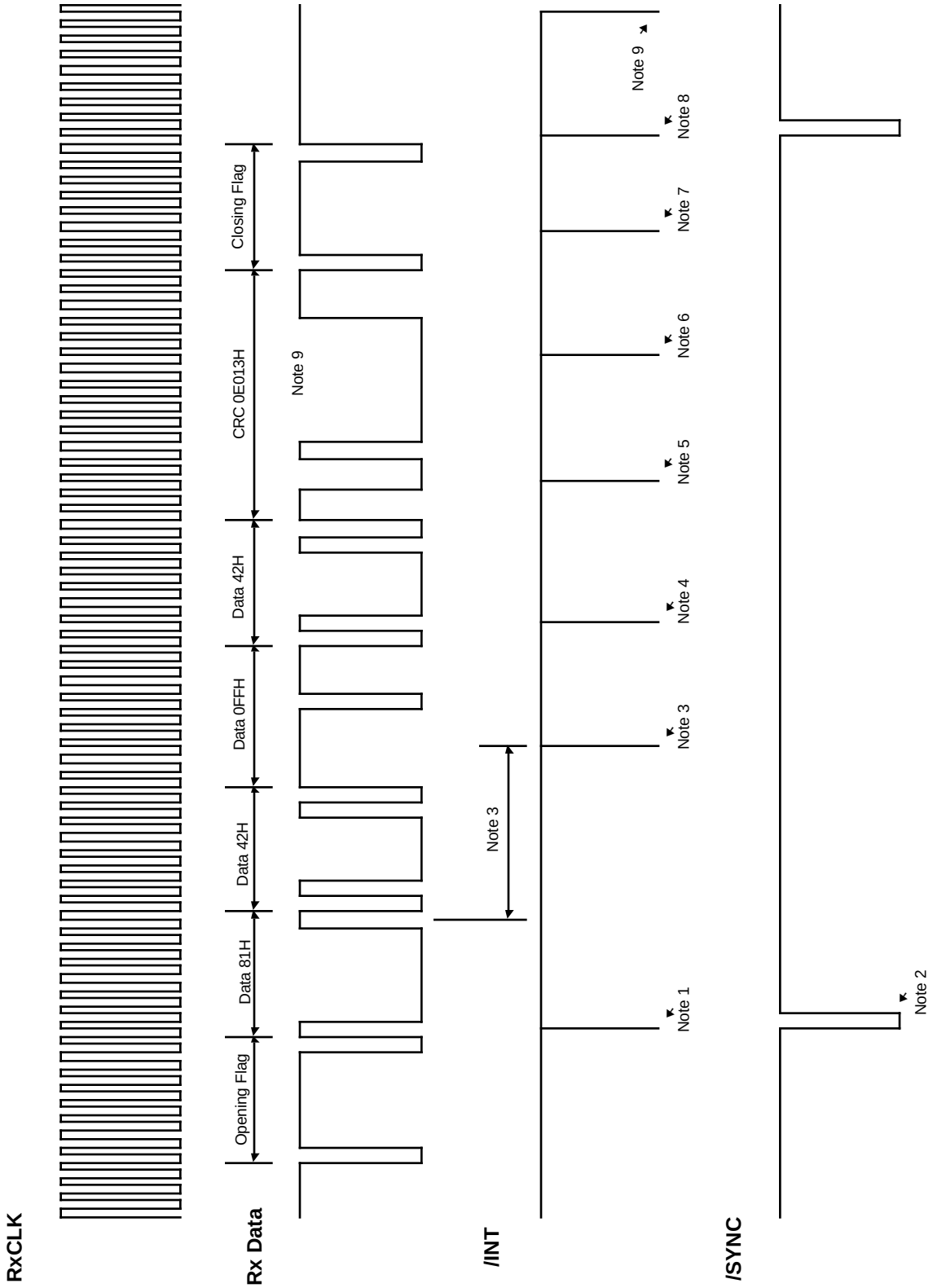



**Figure 2. Typical SDLC Receive Sequence with Receive Interrupts on all Received Characters or Special Condition**

## RECEIVE INTERRUPTS ON FIRST CHARACTER OR SPECIAL CONDITIONS

The sequence of events in this mode is similar to that in “Receive Interrupts on all received characters and Special Conditions”, except that it generates Receive Character Interrupt on the first received character only, and subsequent data is read by the DMA.

The SCC is placed in this mode by programming Bit D4-3 of WR1 to 01. Once programmed in this mode, the SCC generates interrupts when it receives the First Character of the packet or a Special Condition occurs. This mode is for operation with the DMA. On the interrupt for the first received character, DMA is enabled. On Special Conditions (either End-of-Message, overrun, or Parity error, — parity on the SDLC is not normal, however), the service routine stops the DMA and starts over again.

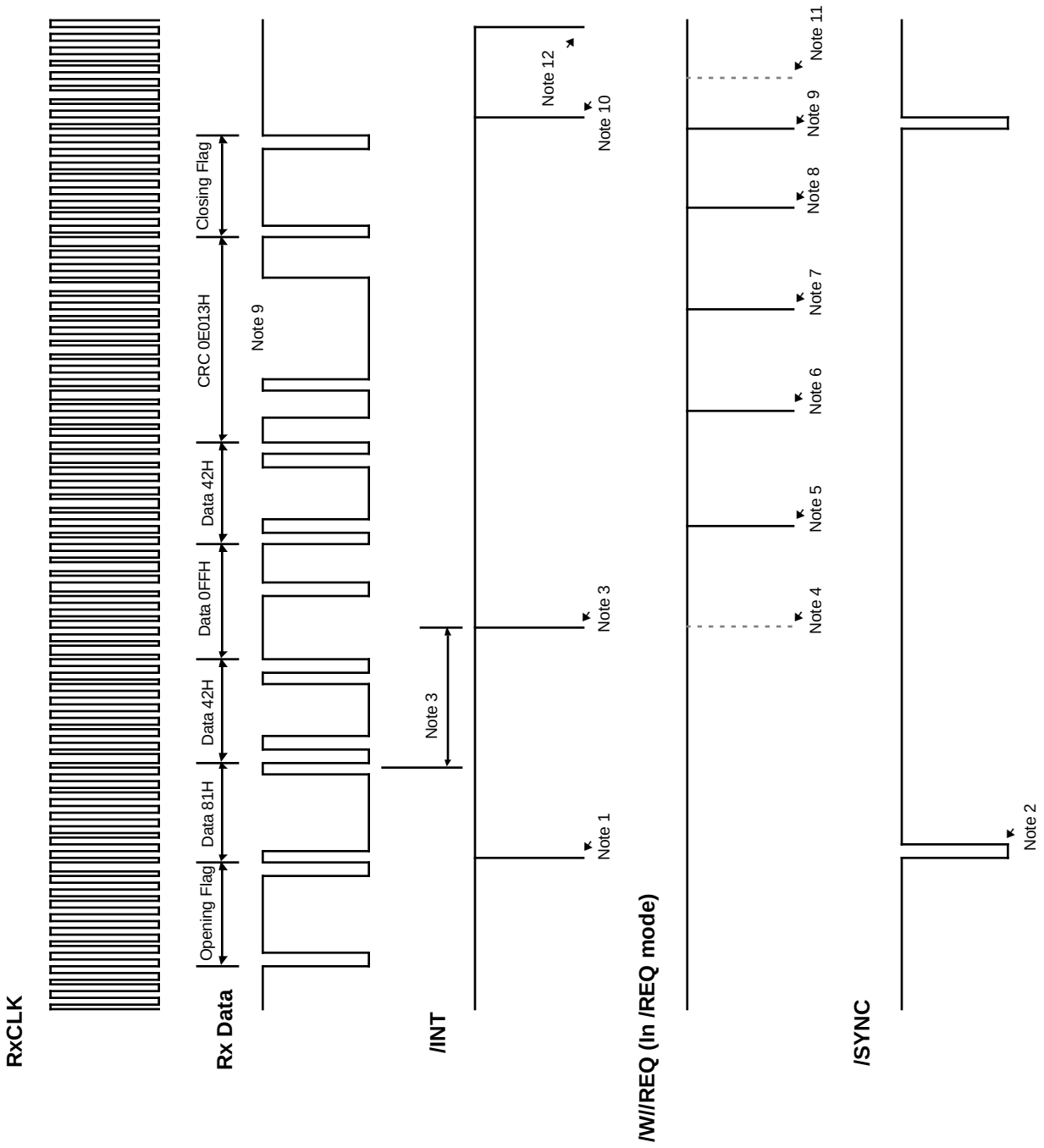



**Figure 3. Typical SDLC Receive Sequence with Receive Interrupts on First Character or Special Condition**

**Notes on Figure 3:**

1. The receiver is usually in hunt mode when waiting for a frame. When the opening flag is received, an External/Status Interrupt is generated, indicating the change from hunt mode to sync mode.

2. The /SYNC output follows the state of the sync register comparison output. The comparison is done on a bit by bit basis, so the /SYNC pin is only active for one bit-time. /SYNC goes active one bit-time after the last bit of the sync character is sampled at the RxD pin.

3. A Receive Character Available Interrupt is generated 11 bit-times after the last bit of the character is sampled at the RxD pin. In this mode, enable the DMA on this interrupt. This interrupt is for data 81H.

4. If SCC's DMA request function has been enabled, /REQ becomes active here.

5. DMA request for data 42H.

6. DMA request for data 0FFH.

7. DMA request for data 42H.

8. DMA request for the first CRC byte. The SCC treats the CRC as data, since the SCC does not yet distinguish a difference between CRC and data!

9. DMA request for the second CRC byte. The closing flag is recognized two bit-times before the second CRC byte is completely assembled in the Receive Shift Register. As soon as it is transferred to the Receive Buffer, it generates a DMA request.

10. This interrupt is EOF (End of Frame), a Special Condition interrupt. This will not occur until the DMA has read the 2nd CRC byte from the Receive Buffer. When it occurs, the Receive Buffer is locked and no more DMA requests can be generated until the Receive Buffer is unlocked by issuing the Error Reset command. Before issuing this command, all of the status bits required (e.g., the CRC error status) must be read, and the last two bytes read by the DMA discarded. The enable interrupt on next Receive Character command must be sent to the SCC so that the next character (i.e., the First Character of the next frame) will produce an interrupt. If this is not done, the character will generate a DMA request, not an interrupt.

    Should a Special Condition occur within the data stream (i.e., for a condition other than EOF) the /INT pin will not go active until the character with the Special Condition has been read by the DMA.

11. DMA request for 2nd CRC bytes. This occurs when the EOF interrupt service routine has not disabled the DMA function of the SCC, and did not read the data after unlocking the buffer by issuing an Error Reset command.

12. External/Status Interrupt for the Sync/Hunt change. This occurs when the SCC recognizes an Abort (Marking line) and forces the receiver into hunt mode. The SCC can be programmed so that the Abort itself generates an interrupt if required. If flag idle was set, this interrupt would not occur.

## RECEIVE INTERRUPTS ON SPECIAL CONDITIONS ONLY

The sequence of event in this mode is similar to that for "Receive Interrupts on first received character or Special Condition," except it will not generate Receive Character Available interrupt at all. This mode is designed for operations where the DMA is pre-programmed, or the application does not have enough time to set up DMA transfer on First Character interrupt.

The SCC is placed in this mode by programming Bit D4-3 of WR1 to 11. Once programmed in this mode, the SCC generates interrupts when Special Conditions occur. On Special Condition (either End-Of-Message or overrun/Parity error, if enabled), corrective action can be taken for that packet.

The SDLC Frame Status Buffer (not available on the NMOS version) is very useful in this mode. First of all, set DMA to transfer several packets. The SDLC Frame Status Buffer holds information which tells you how many bytes were in the received packet and reports whether or not error conditions (overrun/CRC error/parity error) have occurred.

The sequence of events in this mode is identical to the "Receive Interrupts on First Character or Special Condition" mode (Figure 3); Note 3, however, does not apply, and Note 4 should read as follows for this case:

**Note 4 in Receive Interrupts on Special Condition only mode:**
DMA request for data 81H. The DMA function of the SCC should be enabled by this time frame.

## RECEIVING BACK TO BACK FRAME IN RECEIVE INTERRUPTS ON SPECIAL CONDITION ONLY MODE

"Back to Back" frame means there are two frames separated with only one flag — the closing flag of the previous packet also acts as the opening flag of the following packet. Receiving such packets is identical to receiving a single packet, except that the sequence of events happens in a short time around the shared flag.

Assuming SCC is running under Receive Interrupts on Special Condition only mode (under DMA Control), a typical sequence of events is shown in Figure 4. It is identical to that used for "Receive Interrupts on Special Condition Only" mode, with the addition of another following packet.

**Notes on Figure 4:**

1. DMA request data before 0FFH.

2. DMA request for data 0FFH.

3. DMA request for data 42H.

4. DMA request for the first CRC byte. The SCC treats the CRC as data, since the SCC does not yet distinguish a difference between CRC and data!

5. DMA request for the second CRC byte. The closing flag is recognized two bit-times before the second CRC byte is completely assembled in the Receive Shift Register. As soon as it is transferred to the Receive Buffer, it generates a DMA request.

6. This interrupt is EOF (End of Frame), a Special Condition Interrupt. This will not occur until the DMA has read the 2nd CRC byte from the Receive Buffer. When it occurs the Receive Buffer is locked and no more DMA requests can be generated until the Receive Buffer is unlocked by issuing the Error Reset command. Before this command is issued, all of the status bits required (e.g., the CRC error status) must be read, and the last two bytes read by the DMA discarded. The Enable Interrupt on Next Receive Character command must be sent to the SCC so that the next character (i.e., the First Character of the next frame) will produce an interrupt. If this is not done, the character will generate a DMA request, not an interrupt.

   On unlocking the Receive Buffer after the EOF interrupt, no initialization is required with respect to the receiver. All characters have been removed by the DMA and the receiver is ready for the next frame. While the Buffer is locked the SCC can receive 2 7/8 characters (8 bits/character) before there is a danger of the receiver overrunning. The only way that this can be specified is by referencing it to the falling edge of the request for the last CRC byte. This time is a worst case minimum of 33 bit-times (possibly more if there are any characters with inserted zeros). As soon as the Buffer is unlocked an additional 8 (minimum) bit-times become available because the top byte of the Buffer is freed up.

7. DMA request for second CRC byte. This occurs when the EOF interrupt service routine has not disabled the DMA function of the SCC, and fails to read the data after unlocking the FIFO by issuing Error Reset command.

8. DMA request for data 01H.

9. MA request for data 03H.

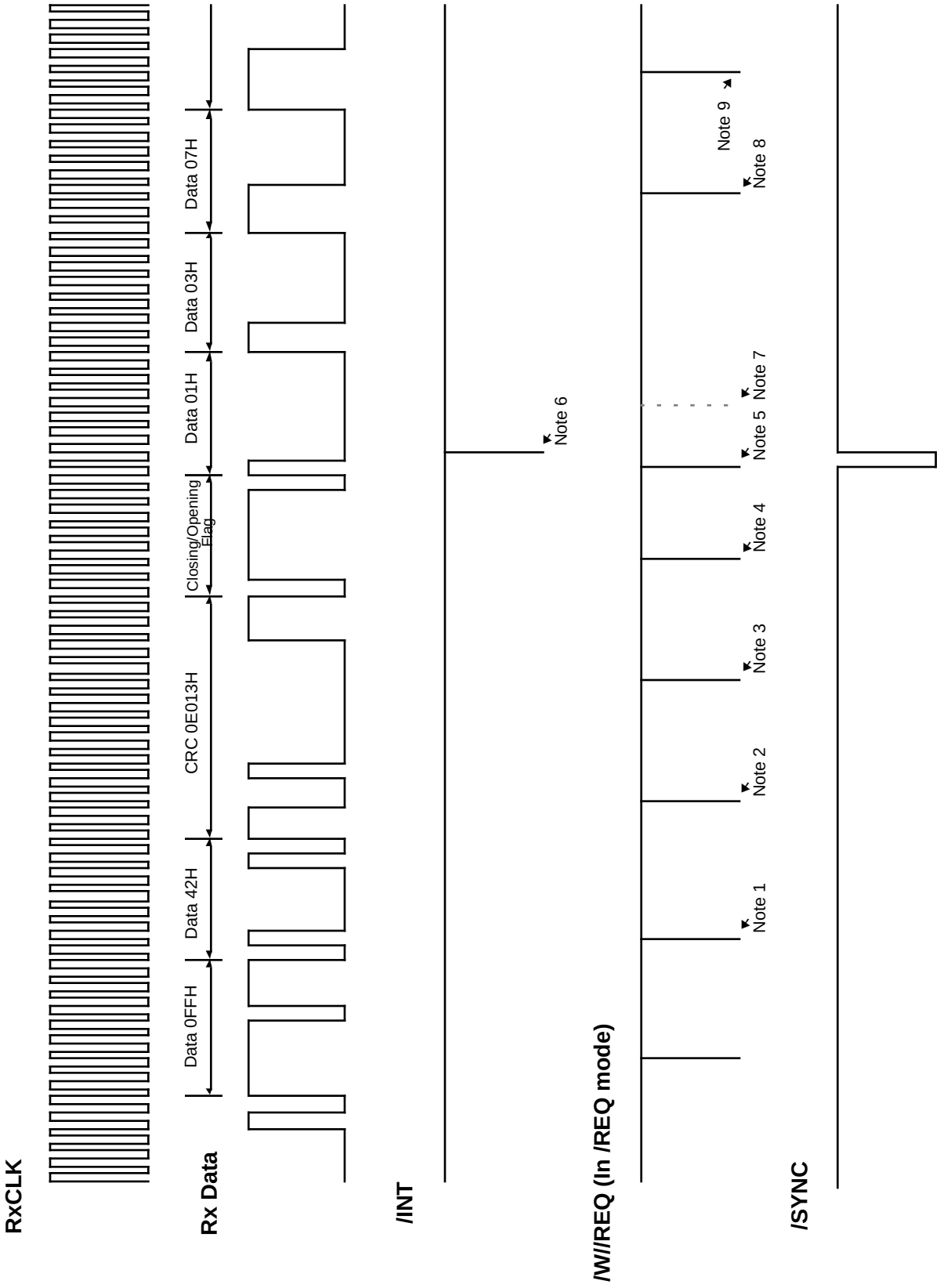



**Figure 4. Receiving "Back to Back" frame with Receive Interrupts on Special Condition Only Mode (DMA Controlled)**

## THE SDLC LOOP MODE

The SDLC Loop mode is one of the protocols used in the ring configuration network topology. The typical network configuration is shown in Figure 5. As shown, there is one Master (or primary) station and several slave (or secondary) stations. This figure does not have a clock connection, but each station's transmit clock must be synchronized to the master SCC. This can be done by feeding the clock using a separate clock line, or by using Phase Locked Loop (PLL) to recover the clock.

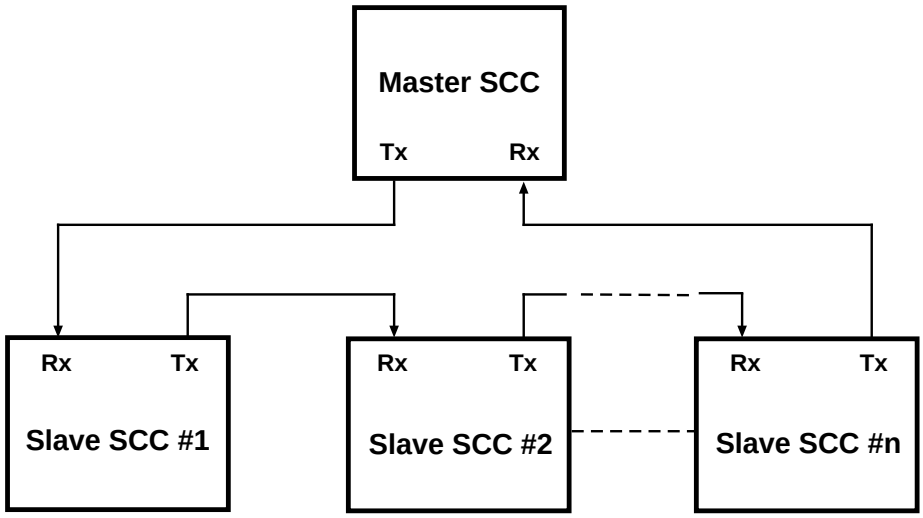



**Figure 5. SDLC Loop Mode Configuration**

This mode is similar to the normal SDLC mode, other than that secondary stations are not allowed to freely send out packets. When a secondary station wants to send a packet, it needs to wait for a special pattern to be received. The pattern is called EOP (End Of Poll), and consists of a 0 followed by seven 1s on the transmission line (as data, it is 11111110). This pattern resembles the SDLC Flag pattern (7EH; 0111110), except the last bit has been changed to a 1 thus turning this pattern into a flag.

Upon network initialization, secondary station TxD and RxD connections use gate propagation delay. On the first EOP, a secondary station inserts one bit -time delay between RxD and TxD, and relays RxD input to TxD. When it has a message to send out, it waits for an EOP. When it detects EOP in this phase, it changes the last bit of the EOP to zero, making it a Flag, then begins to send its own message. From this point on, normal SDLC transmission modes apply. Packets conclude with Mark idle, identifying it as an EOP pattern. The secondary station then reverts back to one bit delay mode.

Figure 6 illustrates this mode's sequence of events. To simplify the example, this figure assumes there is one Master station and one Slave station. If there are more Slave stations, there will be additional one bit time delay per station after the network has initialized for loop mode of operation.

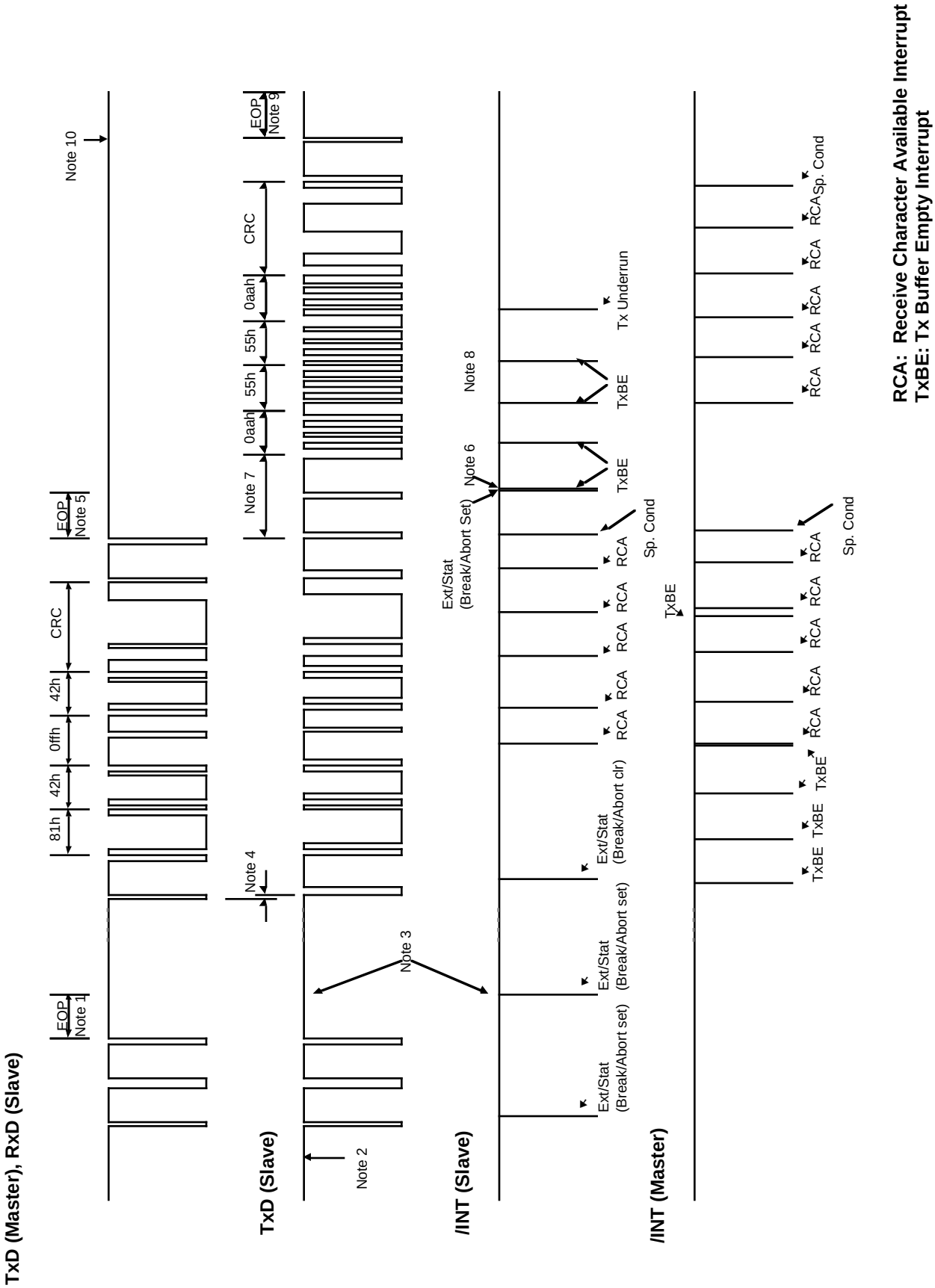



**Figure 6. SDLC Loop Mode**

## THE SDLC LOOP MODE (Continued)

**Notes on Figure 6:**

1. The master SCC sends EOP by switching from flag on idle to mark on idle
2. At initialization, all Slave stations were set up for SDLC loop mode At this point, the Slave station connects its RxD pin to TxD pin with gate propagation delay, and starts to monitor Rx data for the EOP sequence.
3. On receiving the EOP, the slave generates an External/Status Interrupt with Break/Abort bit set. A one bit time delay is inserted between RxD and TxD. (The GAOP,Go active on Poll, bit should be reset at this point to avoid unexpected loop entry by the Slave transmitter.) The Slave's on-loop bit is set and the receiver is in hunt mode.
4. Note that there is a one bit time delay between received data and transmitted data.
5. When the Slave wants to transmit it must first receive an EOP and have GAOP set.
6. On receiving an EOP, the Slave interrupts with Break/Abort clear. The EOP is converted to a flag, the loop sending bit is set, and the transmitter will send flags until data is written into the Transmit Buffer.
7. Note that the flags overlap.
8. When the slave has sent all of its data the GAOP flag should be cleared so that the CRC is sent on underrun.
9. When the closing flag has been sent the Slave reverts to a one bit delay, which produces another EOP.
10. The master must keep its output marking until its receiver has received all frames sent by secondaries.

## CMOS SCC AND ESCC

The discussion above applies to the NMOS SCC and the CMOS SCC without the SDLC Frame Status FIFO feature. The CMOS version and the ESCC have a SDLC Frame Status FIFO for easier handling of the SDLC mode of operation. The SDLC Status FIFO is designed for DMA controlled SDLC receive for high speed SDLC data transmission, or for systems whose CPU interrupt processing is not fast.

This FIFO is able to store up to 10 packets' worth of byte count (14-bit count) and status information (Overrun/Parity/CRC error status). To use this feature, simply enable this FIFO and let DMA transfer data to memory. While DMA is transferring received data to the memory, the CPU will check the FIFO and locate the data in memory, as well as the status information of the received packet.

Other ESCC enhancements make it easier to handle the SDLC mode of operation. These include:

- Deeper FIFO (4 Bytes Transmit, and 8 Bytes receive)
- Automatic Opening Flag transmission
- Automatic EOM reset
- Automatic /RTS deactivation
- Fast /DTR//REQ mode
- Complete CRC reception
- Receive FIFO Antilock feature
- Programmable DMA and interrupt request level
- Improved data setup time specification

For more details on these functions, please refer to the SCC/ESCC Technical manual and related documents.

## CONCLUSION

This application note describes the basic operation of the SCC in SDLC operational modes. With minor variations, most of these operations also apply to the CMOS SCC with Status FIFO enabled and the ESCC.

# USING SCC WITH Z8000 IN SDLC PROTOCOL

## INTRODUCTION

This application note describes the use of the Z8030 Serial Communications Controller (SCC) with the Z8000™ CPU to implement a communications controller in a Synchronous Data Link Control (SDLC) mode of operation. In this application, the Z8002 CPU acts as a controller for the SCC. This application note also applies to the non-multiplexed Z8530.

One channel of the SCC communicates with the remote station in Half Duplex mode at 9600 bits/second. To test this application, two Z8000 Development Modules are used. Both are loaded with the same software routines for initialization and for transmitting and receiving messages. The main program of one module requests the transmit routine to send a message of the length indicated by "COUNT" parameter. The other system receives the incoming data stream, storing the message in its resident memory.

## DATA TRANSFER MODES

The SCC system interface supports the following data transfer modes:

- **Polled Mode.** The CPU periodically polls the SCC status registers to determine if a received character is available, if a character is needed for transmission, and if any errors have been detected.
- **Interrupt Mode.** The SCC interrupts the CPU when certain previously defined conditions are met.
- **Block/DMA Mode.** Using the Wait/Request (/W//REQ) signal, the SCC introduces extra wait cycles in order to synchronize the data transfer between a controller or DMA and the SCC.

The example given here uses the block mode of data transfer in its transmit and receive routines.

## SDLC PROTOCOL

Data communications today require a communications protocol that can transfer data quickly and reliably. One such protocol, Synchronous Data Link Control (SDLC), is the link control used by the IBM Systems Network Architecture (SNA) communications package. SDLC is a subset of the International Standard Organization (ISO) link control called High-Level Data Link Control (HDLC), which is used for international data communications.

SDLC is a bit-oriented protocol (BOP). It differs from byte-control protocols (BCPs), such as Bisync, in that it uses only a few bit patterns for control functions instead of several special character sequences. The attributes of the SDLC protocol are position dependent rather than character dependent, so the data link control is determined by the position of the byte as well as by the bit pattern.

A character in SDLC is sent as an octet, a group of eight bits. Several octets combine to form a message frame, in which each octet belongs to a particular field. Each message contains: opening flag, address, control, information, Frame Check Sequence (FCS), and closing flag (Figure 1).

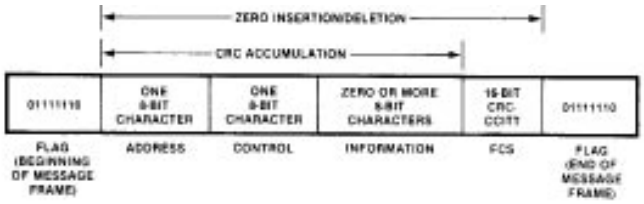



**Figure 1. Fields of the SDLC Transmission Frame**

Both flag fields contain a unique binary pattern, 0111110, which indicates the beginning or the end of the message frame. This pattern simplifies the hardware interface in receiving devices so that multiple devices connected to a common link do not conflict with one another. The receiving devices respond only after a valid flag character has been detected. Once communication is established with a particular device, the other devices ignore the message until the next flag character is detected.

The address field contains one of more octets, which are used to select a particular station on the data link. An address of eight 1s is a global address code that selects all the devices on the data link. When a primary station sends a frame, the address field is used to select one of several secondary stations. When a secondary station sends a message to the primary station, the address field contains the secondary station address, i.e., the source of the message.

The control field follows the address field and contains information about the type of frame being sent. The control field consists of one octet that is always present.

The information field contains any actual transferred data. This field may be empty or it may contain an unlimited number of octets. However, because of the limitations of the error-checking algorithm used in the frame-check sequence, however, the maximum recommended block size is approximately 4096 octets.

The frame check sequence field follows the information or control field. The FCS is a 16-bit Cyclic Redundancy Check (CRC) of the bits in the address, control, and information fields. The FCS is based on the CRC-CCITT code, which uses the polynomial ($x^{16} + x^{12} + x^5 + 1$). The Z8030 SCC contains the circuitry necessary to generate and check the FCS field.

Zero insertion and deletion is a feature of SDLC that allows any data pattern to be sent. Zero insertion occurs when five consecutive 1s in the data pattern are transmitted. After the fifth 1, a 0 is inserted before the next bit is sent. The extra 0 does not affect the data in any way and is deleted by the receiver, thus restoring the original data pattern.

Zero insertion and deletion insures that the data stream will not contain a flag character or abort sequence. Six 1s preceded and followed by 0s indicate a flag sequence character. Seven to fourteen 1s signify and abort; Seven to fourteen 1s signify an abort; 15 or more 1s indicate an idle (inactive) line. Under these three conditions, zero insertion and deletion are inhibited. Figure 2 illustrates the various line conditions.

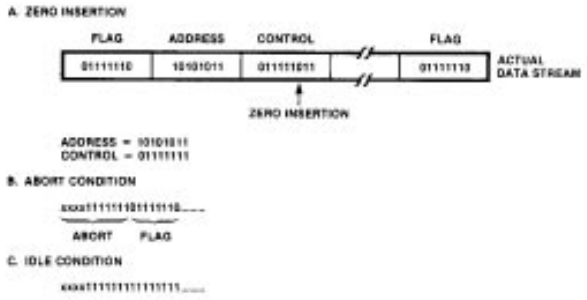



**Figure 2. Bit Patterns for Various Line Conditions**

The SDLC protocol differs from other synchronous protocols with respect to frame timing. In Bisync mode, for example, a host computer might temporarily interrupt transmission by sending sync characters instead of data. This suspended condition continues as long as the receiver does not time out. With SDLC, however, it is invalid to send flags in the middle of a frame to idle the line. Such action causes an error condition and disrupts orderly operation. Thus, the transmitting device must send a complete frame without interruption. If a message cannot be transmitted completely, the primary station sends an abort sequence and restarts the message transmission at a later time.

## SYSTEM INTERFACE

The Z8002 Development Module consists of a Z8002 CPU, 16K words of dynamic RAM, 2K words of EPROM monitor, a Z80A SIO providing dual serial ports, a counter/timer channels, two Z80A PIO devices providing 32 programmable I/O lines, and wire wrap area for prototyping. The block diagram is depicted in Figure 3. Each of the peripherals in the development module is connected in a prioritized daisy chain configuration. The SCC is included in this configuration. The SCC is included in this configuration by tying its IEI line to the IEO line of another device, thus making it one step lower in interrupt priority compared to the other device.

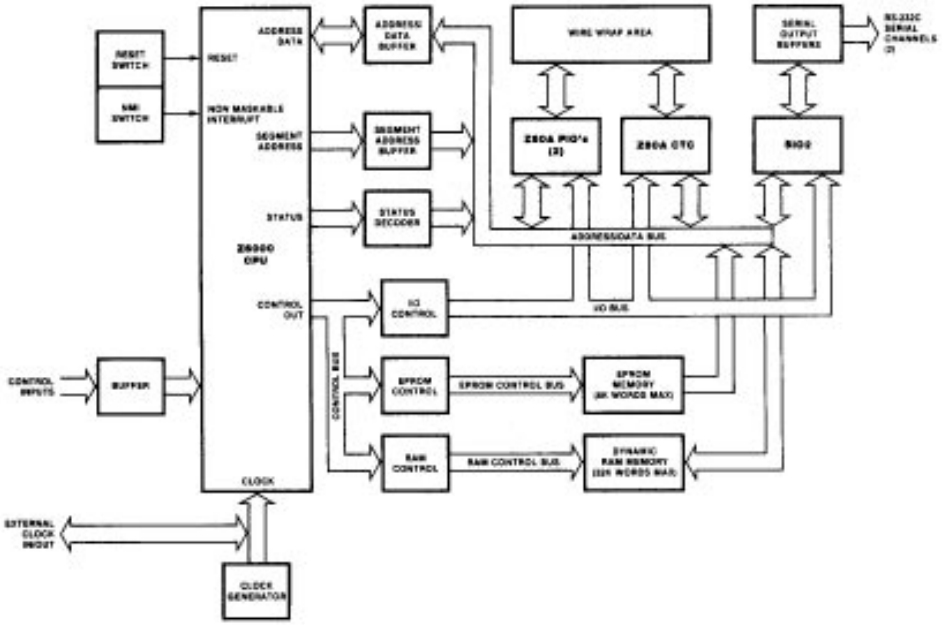



**Figure 3. Block Diagram of Z8000 DM**

## SYSTEM INTERFACE (Continued)

Two Z8000 Development Modules containing SCCs are connected as shown in Figure 4 and Figure 5. The Transmit Data pin of one is connected to the Receive Data pin of the other and vice versa. The Z8002 is used as a host CPU for loading the modules; memories with software routines.

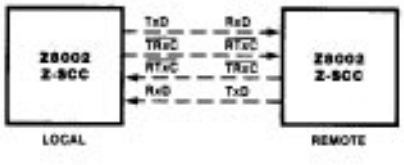



**Figure 4. Block Diagram of Two Z8000 CPUs**

The Z8002 CPU can address either of the two bytes contained in 16-bit words. The CPU uses an even address (16 bits) to access the most significant byte of a word and an odd address for the least significant byte of a word.

When the Z8002 CPU uses the lower half of the Address/Data bus (AD7-AD0 the least significant byte) for byte read and write transactions during I/O operations, these transactions are performed between the CPU and I/O ports located at odd I/O addresses. Since the SCC is attached to the CPU on the lower half of the A/D bus, its registers must appear to the CPU at odd I/O addresses. To achieve this, the SCC can be programmed to select its internal registers using lines AD5-AD1. This is done either automatically with the Force Hardware Reset command in WR9 or by sending a Select Shift Left Mode command to WR0B in channel B of the SCC. For this application, the SCC registers are located at I/O port address “Fexx”. The Chip Select signal (/CSO) is derived by decoding I/O address “FE” hex from lines AD15-AD8 of the controller.

To select the read/write registers automatically, the SCC decodes lines AD5-AD1 in Shift Left mode. The register map for the SCC is depicted in Table 1.

**Table 1. Register Map**

| Address (Hex) | Write Register | Read Register |
|---|---|---|
| FE01 | WR0B | RR0B |
| FE03 | WR1B | RR1B |
| FE05 | WR2 | RR2B |
| FE07 | WR3B | RR3B |
| FE09 | WR4B | |
| FE0B | WR5B | |
| FE0D | WR6B | |
| FE0F | WR7B | |
| FE11 | B DATA | B DATA |
| FE13 | WR9 | |
| FE15 | WR10B | RR10B |
| FE17 | WR11B | |
| FE19 | WR12B | RR12B |
| FE1B | WR13B | RR13B |
| FE1D | WR14B | |
| FE1F | WR15B | RR15B |
| FE21 | WR0A | RR0A |
| FE23 | WR1A | RR1A |
| FE25 | WR2 | RR2A |
| FE27 | WR3A | RR3A |
| FE29 | WR4A | |
| FE2B | WR5A | |
| FE2D | WR6A | |
| FE2F | WR7A | |
| FE31 | A DATA | A DATA |
| FE33 | WR9 | |
| FE35 | WR10A | RR10A |
| FE37 | WR11A | |
| FE39 | WR12A | RR12A |
| FE3B | WR13A | RR13A |
| FE3D | WR14A | |
| FE3F | WR15A | RR15A |

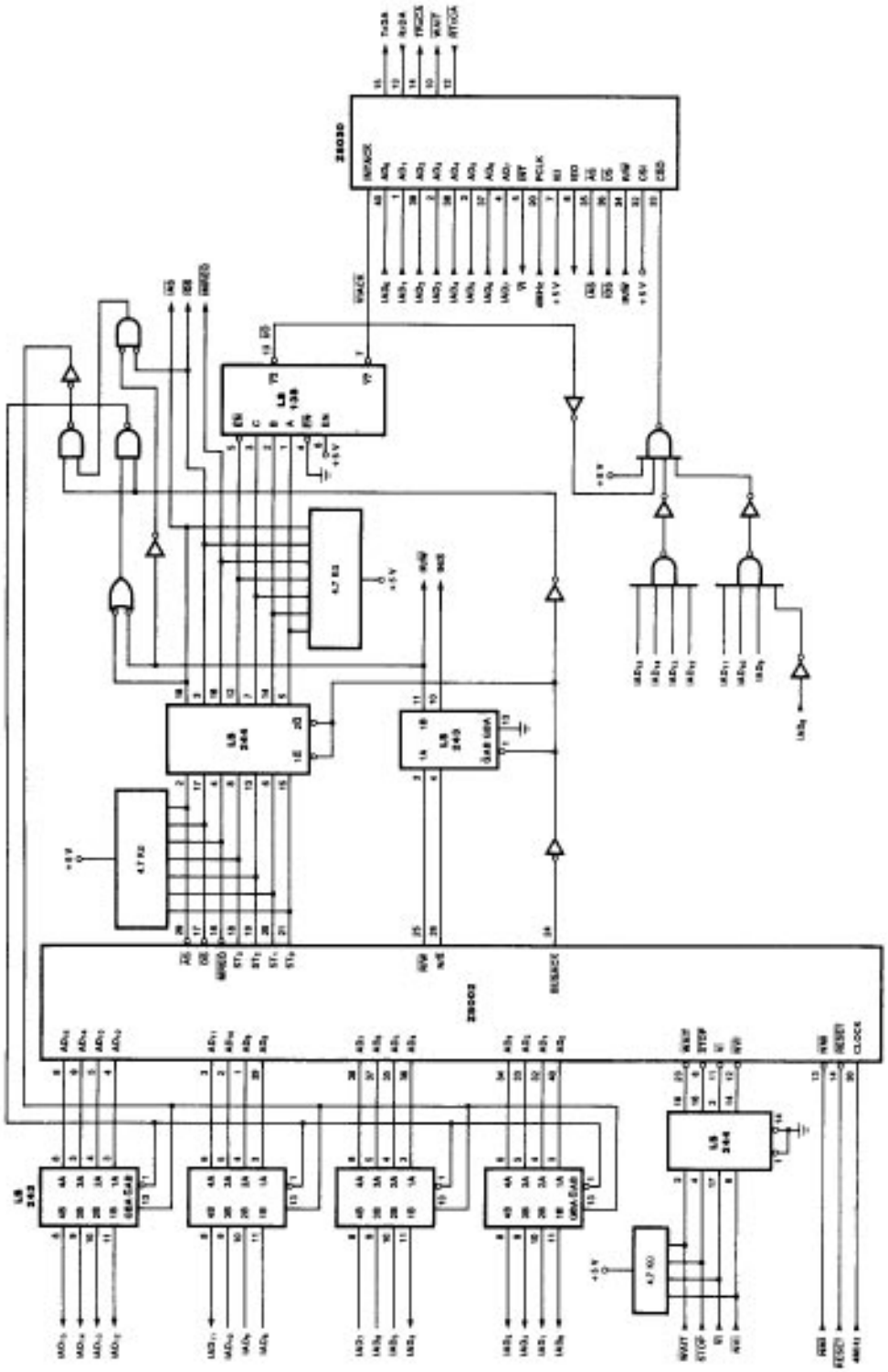


**Figure 5. Z8002 With SCC**

## INITIALIZATION

The SCC can be initialized for use in different modes by setting various bits in its write registers. First, a hardware reset must be performed by setting bits 7 and 6 of WR9 to one; the rest of the bits are disabled by writing a logic zero.

SDLC protocol is established by selecting a SDLC mode, sync mode enable, and a x1 clock in WR4. A data rate of 9600 baud, NRZ encoding, and a character length of eight bits are among the other options that are selected in this example (Table 2).

Note that WR9 is accessed twice, first to perform a hardware reset and again at the end of the initialization sequence to enable the interrupts. The programming sequence depicted in Table 2 establishes the necessary parameters for the receiver and transmitter so that they are ready to perform communication tasks when enabled.

**Table 2. Programming Sequence for Initialization**

| Register | Value (Hex) | Effect |
|---|---|---|
| WR9 | C0 | Hardware reset |
| WR4 | 20 | x1 clock, SDLC mode, sync mode enable |
| WR10 | 80 | NRZ, CRC preset to one |
| WR6 | AB | Any station address e.g. “AB” |
| WR7 | 7E | SDLC flag (01111110) = “7E” |
| WR2 | 20 | Interrupt vector “20” |
| WR11 | 16 | Tx clock from BRG output, /TRxC pin = BRG out |
| WR12 | CE | Lower byte of time constant = “CE” for 9600 baud |
| WR13 | 0 | Upper byte = 0 |
| WR14 | 03 | BRG source bit =1 for PCKL as input, BRG enable |
| WR15 | 00 | External Interrupt Disable |
| WR5 | 60 | Transmit 8 bits/character SDLC CRC |
| WR3 | C1 | Rx 8 bits/character, Rx enable (Automatic Hunt mode) |
| WR1 | 08 | ext int. disable |
| WR9 | 09 | MIE, VIS, status Low |

The Z8002 CPU must be operated in System mode to execute privileged I/O instructions. So the Flag and Control Word (FCW) should be loaded with system normal (S//N), and the Vectored Interrupt Enable (VIE) bits set. The Program Status Area Pointer (PSAP) is loaded with address %4400 using the Load Control Instruction (LDCTL). If the Z8000 Development Module is intended to be used, the PSAP need not be loaded by the programmer because the development module’s monitor loads it automatically after the NMI button is pressed.

Since VIS and Status Low are selected in WR9, the vectors listed in Table 3 will be returned during the Interrupt Acknowledge cycle. Of the four interrupts listed, only two, Ch A Receive Character Available and Ch A Special Receive Condition, are used in the example given here.

**Table 3. Interrupt Vectors**

| Vector (Hex) | PS Address | Interrupt |
|---|---|---|
| 28 | 446E | Ch A Transmit Buffer Empty |
| 2A | 4472 | Ch A External Status Change |
| 2C | 4476 | Ch A Receive Char. Available |
| 2E | 447A | Ch A Special Receive Condition |

* Assuming that PSAP has been set to 4400 hex, “PS Address” refers to the location in the Program Status Area where the service routine address is stored for that particular interrupt.

## TRANSMIT OPERATION

To transmit a block of data, the main program calls up the transmit data routine. With this routine, each message block to be transmitted is stored in memory, beginning with location “TBUF” The number of characters contained in each block is determined by the value assigned to the “COUNT” parameter in the main module.

To prepare for transmission, the routine enables the transmitter and selects the Wait On Transmit function; it then enables the wait function. The Wait on Transmit function indicates to the CPU whether or not the SCC is ready to accept data from the CPU. If the CPU attempts to send data to the SCC when the transmit buffer is full, the SCC asserts its /WAIT line and keeps it Low until the buffer is empty. In response, the CPU extends its I/O cycles until the /WAIT line goes inactive, indicating that the SCC is ready to receive data.

The CRC generator is reset and the Transmit CRC bit is enabled before the first character is sent, thus including all the characters sent to the SCC in the CRC calculation.

The SCC transmit underrun/EOM latch must be reset sometime after the first character is transmitted by writing a Reset Tx Underrun/EOM command to WR0. When this latch is reset, the SCC automatically appends the CRC characters to the end of the message in the case of an underrun condition.

Finally, a three-character delay is introduced at the end of the transmission, which allows the SCC sufficient time to transmit the last data byte and two CRC characters before disabling the transmitter.

## RECEIVE OPERATION

Once the SCC is initialized, it can be prepared to receive the message. First, the receiver is enabled, placing the SCC in Hunt mode and thus setting the Sync/Hunt bit in status register RR0 to 1. In Hunt mode, the receiver searches the incoming data stream for flag characters. Ordinarily, the receiver transfers all the data received between flags to the receive data FIFO. If the receiver is in Hunt mode, however, no data transfer takes place until an opening flag is received. If an abort sequence is received, the receiver automatically re-enters Hunt mode. The Hunt status of the receiver is reported by the Sync/Hunt bit in RR0.

The second byte of an SDLC frame is assumed by the SCC to be the address of the secondary stations for which the frame is intended. The SCC provides several options for handling this address. If the Address Search Mode bit D2 in WR3 is set to zero, the address recognition logic is disabled and all the received data bytes are transferred to the receive data FIFO. In this mode, software must perform any address recognition. If the Address Search Mode bit is set to one, only those frames with addresses that match the address programmed in WR6 or the global address (all 1s) will be transferred to the receive data FIFO. If the Sync Character Load Inhibit bit (D1) in WR3 is set to zero, the address comparison is made across all eight bits of WR6. The comparison can be modified so that only the four most significant bits of WR6 need match the received address. This alterations made by setting the Sync Character Load Inhibit bit to one. In this mode, the address field is still eight bits wide and is transferred to the FIFO in the same manner as the data. In this application, the address search is performed.

When the address match is accomplished, the receiver leaves the Hunt mode and establishes the Receive Interrupt on First Character mode. Upon detection of the receive interrupt, the CPU generates an Interrupt Acknowledge Cycle. The SCC returns the programmed vector %2C. This vector points to the location %4472 in the Program Status Area which contains the receive interrupt service routine address.

The receive data routine is called from within the receive interrupt service routine. While expecting a block of data, the Wait on Receive function is enabled. Receive read buffer RR8 is read and the characters are stored in memory location RBUF. The SCC in SDLC mode automatically enables the CRC checker for all data between opening and closing flags and ignores the Receive CRC Enable bit (D3) in WR3. The result of the CRC calculation for the entire frame in RR1 becomes valid only when the End of Frame bit is set in RR1. The processor does not use the CRC bytes, because the last two bits of the CRC are never transferred to the receive data FIFO and are not recoverable.

When the SCC recognizes the closing flag, the contents of the Receive Shift register are transferred to the receive data FIFO, the Residue Code (not applicable in this application) is latched, the CRC error bit is latched in the status FIFO, and the End of Frame bit is set in the receive status FIFO, a special receive condition interrupt occurs. The special receive condition register RR1 is read to determine the bit is zero, the frame received is assumed to be correct; if the bit is 1, an error in the transmission is indicated.

Before leaving the interrupt service routine, Reset Highest IUS (Interrupt Under Service), Enable Interrupt on Next Receive Character, and Enter Hunt Mode commands are issued to the SCC.

## RECEIVE OPERATION (Continued)

If receive overrun error is made, a special condition interrupt occurs. The SCC presents vector %2E to the CPU, and the service routine located at address %447A is executed. Register RR1 is read to determine which error occurred. Appropriate action to correct the error should be taken by the user at this point. Error Reset and Reset Highest IUS commands are given to the SCC before returning to the main program so that the other low-priority interrupts can occur.

In addition to searching the data stream for flags, the receiver also scans for seven consecutive 1s, which indicates an abort condition. This condition is reported in the Break/Abort bit (D7) in RR0. This is one of many possible external status conditions. As a result transitions of this bit can be programmed to cause an external status interrupt. The abort condition is terminated when a zero is received, either by itself or as the leading zero of a flag. The receiver leaves Hunt mode only when a flag is found.

## SOFTWARE

Software routines are presented in the following pages. These routines can be modified to include various other options (e.g., SDLC Loop, Digital Phase Locked Loop etc.). By modifying the WR10 register, different encoding methods (e.g., NRZI, FM0, FM1) other than NRZ can be used.

# Appendix

Software Routines

```
plzasm   1.3
LOC    OBJ CODE     STMT SOURCE STATEMENT

                      1
                      2
                      3          SDLC MODULE
                        $LISTON $TTY
                        CONSTANT
                        WRSA     :=  %FE21              !BASE ADDRESS FOR WR0 CHANNEL A!
                        RRSA     :=  %FE21              !BASE ADDRESS FOR RR0 CHANNEL A!
                        RBUF     :=  %5400              !BUFFER AREA FOR RECEIVE CHARACTER!
                        PSAREA   :=  %4400              !START ADDRESS FOR PROGRAM STAT AREA!
                        COUNT    :=   12                !NO. OF CHAR. FOR TRANSMIT ROUTINE!
0000                    GLOBAL MAIN PROCEDURE
                        ENTRY

0000 7601                       LDA     R1,PSAREA
0002 4400
0004 7D1D                       LDCTL   PSAPOFF,R1      !LOAD PSAP!
0006 2100                       LD      R0,#%5000
0008 5000
000A 3310                       LD      R1(#%1C),R0     !FCW VALUE(%5000) AT %441C FOR VECTORED!
000C 001C                                               !INTERRUPTS!
000E 7600                       LDA     R0,REC
0010 00D6'
0012 3390                       LD      R1(#%76),R0     !EXT. STATUS SERVICE ADDR. AT %4476 IN!
0014 0076                                               !PSA!
0016 7600                       LDA     R0,SPCOND
0018 00FA'
001A 3310                       LD      R1(#%7A),R0     !SP.COND.SERVICE ADDR AT %447A IN PSA!
001C 007A
001E 5F00                       CALL    INIT
0020 0034'
0022 5F00                       CALL    TRANSMIT
0024 008C'
0026 EBFF                       JR      $

0028 AB                 TBUF:   BVAL    %AB             !STATION ADDRESS!
0029 48                         BVAL    'H'
002A 45                         BVAL    'E'
002B 4C                         BVAL    'L'
002C 4C                         BVAL    'L'
002D 4F                         BVAL    'O'
002E 20                         BVAL    ' '
002F 54                         BVAL    'T'
0030 48                         BVAL    'H'
0031 45                         BVAL    'E'
0032 52                         BVAL    'R'
0033 45                         BVAL    'E'

0034                            END     MAIN
```

## **SOFTWARE** (Continued)

```
                ;****************** INITIALISATION ROUTINE FOR Z-SCC ***********************;

0034                    GLOBAL  INIT PROCEDURE
                        ENTRY
0034 2100               LD      R0,#15          !NO.OF PORTS TO WRITE TO!
0036 00DF
0038 7602               LDA     R2,SCCTAB       !ADDRESS OF DATA FOR PORTS!
003A 004E'
003C 2101       ALOOP:  LD      R1,#WR0A
003E FE21
0040 0029               ADDB    RL1,@R2
0042 A920               INC     R2
0044 3A22               OUTIB   @R1,@R2,R0      !POINT TO WR0A,WR1A ETC THRO LOOP!
0046 0018
0048 8D04               TEST    R0              !END OF LOOP?!
004A EEFB               JR      NE,ALOOP        !NO,KEEP LOOPING!
004C 9E08               RET
004E 12         SCCTAB: BVAL    2*9
004F C0                 BVAL    %C0             !WR9=HARDWARE RESET!
0050 08                 BVAL    2*4
0051 20                 BVAL    %20             !WR4=X1 CLK,SDLC,SYNC MODE!
0052 14                 BVAL    2*10
0053 80                 BVAL    %80             !WR10=CRC PRESET ONE,NRZ,FLAG ON IDLE,!
                                                      !FLAG ON UNDERRUN!
0054 0C                 BVAL    2*6
0055 AB                 BVAL    %AB             !WR6= ANY ADDRESS FOR SDLC STATION!
0056 0E                 BVAL    2*7
0057 7E                 BVAL    %7E             !WR7=SDLC FLAG CHAR!
0058 04                 BVAL    2*2
0059 20                 BVAL    %20             !WR2=INT VECTOR %20!
005A 16                 BVAL    2*11
005B 16                 BVAL    %16             !WR11=Tx CLOCK & TRxC OUT=BRG OUT!
005C 18                 BVAL    2*12
005D CE                 BVAL    %CE             !WR12= LOWER TC=CE!
005E 1A                 BVAL    2*13
005F 00                 BVAL    0               !WR13= UPPER TC=0!
0060 1C                 BVAL    2*14
0061 03                 BVAL    %03             !WR14=BRG ON,BRG SRC=PCLK!
0062 1E                 BVAL    2*15
0063 00                 BVAL    %00             !WR15=EXT INT. DISABLE!
0064 0A                 BVAL    2*5
0065 60                 BVAL    %60             !WR5=Tx 8 BITS/CHAR, SDLC CRC!
0066 06                 BVAL    2*3
0067 C5                 BVAL    %C5             !WR3=ADDR SRCH,REC ENABLE!
0068 02                 BVAL    2*1
0069 08                 BVAL    %08             !WR1=RX INT ON 1ST & SP COND,!
                                                      !EXT INT DISABLE!
006A 12                 BVAL    2*9
006B 09                 BVAL    %09             !WR9= NIE,VIS,STATUS LOW!
006C            END     INIT


                ;****************** RECEIVE ROUTINE ************************************;

                ;                   RECEIVE A BLOCK OF MESSAGE                          !

006C                    GLOBAL  RECEIVE PROCEDURE
                        ENTRY
006C C828               LDB     RL0,#%28        !WAIT ON RECV.!
006E 3A86               OUTB    WR0A+2,RL0
0070 FE23
0072 6008               LDB     RL0,%A8
0074 00A8
0076 3A86               OUTB    WR0A+2,RL0      !ENABLE WAIT FNC. SP. COND. INT!
0078 FE23
007A 2101               LD      R1,#RR0A+16
007C FE31
007E 2102               LD      R2,#COUNT+2     !COUNT+2 CHARACTERS TO READ!
0080 000E
0082 2103               LD      R3,#RBUF        !RECEIVE BUFFER IN MEMORY!
0084 5400
0086 3A18               INDRB   @R3,@R1,R2      !READ THE ENTIRE MESSAGE!
0088 0230
008A 9E08               RET
008C            END RECEIVE
```

```
                        !***************  TRANSMIT ROUTINE ***********************************!
                        !                 SEND A BLOCK OF EIGHT  DATA CHARACTERS               !
                        !                 THE BLOCK STARTS AT LOCATION TBUF                    !

008C                            GLOBAL  TRANSMIT PROCEDURE
                                ENTRY
008C 2102                               LD      R2,#TBUF        !PTR TO START OF BUFFER!
008E 0028'
0090 C868                               LDB     RL0,#%68
0092 3A86                               OUTB    WR0A+10,RL0     !ENABLE TRANSMITTER!
0094 FE2B
0096 C800                               LDB     RL0,#%00        !WAIT ON TRANSMIT!
0098 3A86                               OUTB    WR0A+2,RL0
009A FE23
009C C888                               LDB     RL0,#%88
009E 3A86                               OUTB    WR0A+2,RL0      !WAIT ENABLE!
00A0 FE23
00A2 C880                               LDB     RL0,#%80
00A4 3A86                               OUTB    WR0A,RL0        !RESET TxCRC GENERATOR!
00A6 FE21
00A8 2101                               LD      R1,#WR0A+16     !WR8A SELECTED!
00AA FE31
00AC 2100                               LD      R0,#1
00AE 0001
00B0 C869                               LDB     RL0,#%69        !SDLC CRC!
00B2 3A86                               OUTB    WR0A+10,RL0     !WR5A-TxCRC ENABLE!
00B4 FE2B
00B6 3A22                               OTIRB   @R1,@R2,R0      !SEND ADDRESS!
00B8 0010
00BA C8C0                               LDB     RL0,#%C0
00BC 3A86                               OUTB    WR0A,RL0        !RESET TxUND/EOM LATCH!
00BE FE21
00C0 2100                               LD      R0,#COUNT-1
00C2 000B
00C4 3A22                               OTIRB   @R1,@R2,R0      !SEND MESSAGE!
00C6 0010
00C8 2100                               LD      R0,#926         !CREATE DELAY BEFORE DISABLING!
00CA 039E
00CC F0B1                       DEL:    DJNZ    R0,DEL          !TRANSMITTER SO THAT CRC CAN BE!
00CE C8D0                               LDB     RL0,#0          !SENT!
00D0 3A86                               OUTB    WR0A+10,RL0     !DISABLE TRANSMITTER!
00D2 FE2B
00D4 9ED8                               RET
00D6                            END TRANSMIT


                        !************** RECEIVE INT. SERVICE ROUTINE ************************!

00D6                              GLOBAL REC PROCEDURE
                                  ENTRY
00D6 93F3                               PUSH    @R15,R3
00D8 93F2                               PUSH    @R15,R2
00DA 93F1                               PUSH    @R15,R1
00DC 93F0                               PUSH    @R15,R0
00DE 3A84                               INB     RL1,RR0A        !READ STATUS REG RR0A!
00E0 FE21
00E2 A690                               BITB    RL1,#0          !TEST IF Rx CHAR SET!
00E4 E602                               JR      Z,RESET         !YES CALL RECEIVE ROUTINE!
00E6 5FD0                               CALL    RECEIVE
00E8 006C'
00EA C838                       RESET:  LDB     RL0,#%38
00EC 3A86                               OUTB    WR0A,RL0        !RESET HIGHEST IUS!
00EE FE21
00F0 97F0                               POP     R0,@R15
00F2 97F1                               POP     R1,@R15
00F4 97F2                               POP     R2,@R15
00F6 97F3                               POP     R3,@R15
00F8 7B00                               IRET
00FA                            END REC
```

## **RECEIVE OPERATION** (Continued)

```
                    |************ SPECIAL CONDITION INTERRUPT SERVICE ROUTINE ***************|

00FA                        GLOBAL SPCOND PROCEDURE
                            ENTRY

00FA 93F0                           PUSH    @R15,R0
00FC 3A84                           INB     RL0,RRDA+2      |READ ERRORS|
00FE FE23
0100 A607                           BITB    RL0,#7          |END OF FRAME ?|
                            |PROCESS OVERRUN,FRAMING ERRORS IF ANY|
0102 E603                           JR      Z,RESE
0104 C820                           LDB     RL0,#%20
0106 3A86                           OUTB    WRDA,RL0        | YES,ENABLE INT ON NEXT RBC CHAR|
0108 FE21
010A C830                   RESE:   LDB     RL0,#%30
010C 3A86                           OUTB    WRDA,RL0        |ERROR RESET|
010E FE21
0110 C808                           LDB     RL0,#%08
0112 3A86                           OUTB    WRDA+2,RL0      |WAIT DISABLE,RxINT ON 1ST OR SP COND.|
0114 FE23
0116 C838                           LDB     RL0,#%38
0118 3A86                           OUTB    WRDA,RL0        |RESET HIGHEST IUS|
011A FE21
011C 97F0                           POP     R0,@R15
011E 7B00                           IRET

0120                        END SPCOND

                            END SDLC
```

# BOOST YOUR SYSTEM PERFORMANCE USING THE ZILOG ESCC™

*For greater testability, larger interface flexibility, and increased CPU/DMA offloading, replace the SCC with the ESCC™ Controller... and utilize the ESCC to its full potential.*

## INTRODUCTION

This App Note (Application Note) describes the differences between the SCC (Z8030/8530, Z80C30/85C30) and ESCC (Z80230/85230). It outlines the procedures in utilizing the ESCC to its full potential. Application details such as Schematics and Program Listings are not included since these materials are in our various application support products.

**Note:** The author assumes the audience has fundamental Datacommunications knowledge and basic familiarity with Zilog SCC products.

**Notes:** All Signals with a preceding front slash, "/", are active Low, e.g.: B//W (WORD is active Low); /B/W (BYTE is active Low, only).

Power connections follow conventional descriptions below:

| Connection | Circuit | Device |
|---|---|---|
| Power | $V_{CC}$ | $V_{DD}$ |
| Ground | GND | $V_{SS}$ |

## ESCC/SCC DIFFERENCES

The differences between the ESCC and SCC are shown below:

| ESCC ENHANCEMENT | PERFORMANCE BENEFITS |
|---|---|
| 1. Extended Read Enable of Write Registers | - Improves Testability<br>- Ability to examine SDLC status on-the-fly |
| 2. Hardware Improvement | |
| - Modified WRITE Timing | - Improves Interface to 80X86 CPU |
| - Modified DMA Request on | - Improves Interface DMA-driven system |
| - Transmit Deactivation Timing | |
| 3. Throughput improvement | |
| - Deeper Transmit FIFO | - Reduces TBE Interrupt Frequency by 3/4 |
| - Deeper Receive FIFO | - Reduces RCA Interrupt Frequency by 3/4 |
| - FIFO Interrupt Level | - Flexibility in Adapting CPU Workload |
| 4 SDLC End Of Frame Improvement | |
| - Automatic RTS Deassertion after Closing Flag | - Reduces CPU and DMA Controller Overhead after End Of Frame |
| - Automatic Opening Flag Transmission | |
| - Automatic TxD Forced High in SDLC with NRZI Encoding When Marking Idle After End Of Frame | - Allows Optimal SDLC Line Utilization |
| - Improvement to Allow Transmission of Back-to-Back Frames with a Shared Flag | |
| - Status FIFO Anti-Lock Feature in DMA-Driven System | |

The differences between the ESCC and SCC are summarized by a new register, WR7' (Figure 1).

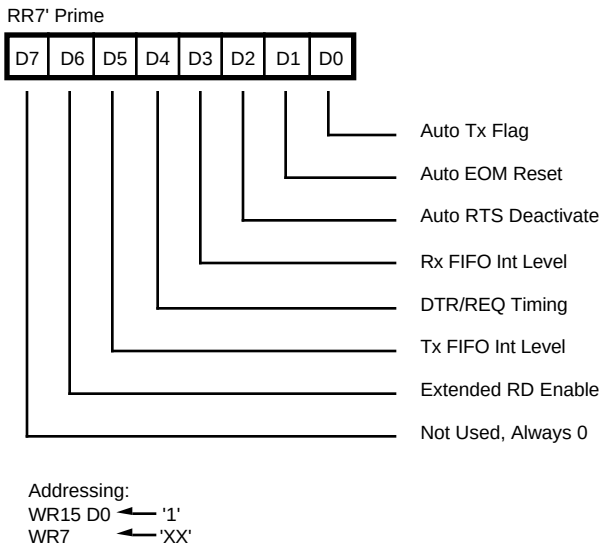



**Figure 1. WR7' Definition**

The advantages of the new features are illustrated in the following examples.

One of the features that is offered by the ESCC, but not the SCC, is Extended Read Enable. Write Register values from the WR3, WR4, WR5, WR7', and WR10 can be examined in the ESCC but not the SCC. This feature improves system testability. It is also crucial for SCC/ESCC differentiation and allows generic software structures for all SCC/ESCC devices.

Flowchart 1 (Figure 2) shows a generic software structure applicable for all SCC/ESCC initializations. Flowchart 2 (Figure 3) suggests a method for determining which type of SCC/ESCC™ device is in the socket. This software structure helps the development of software drivers independent of the device type.

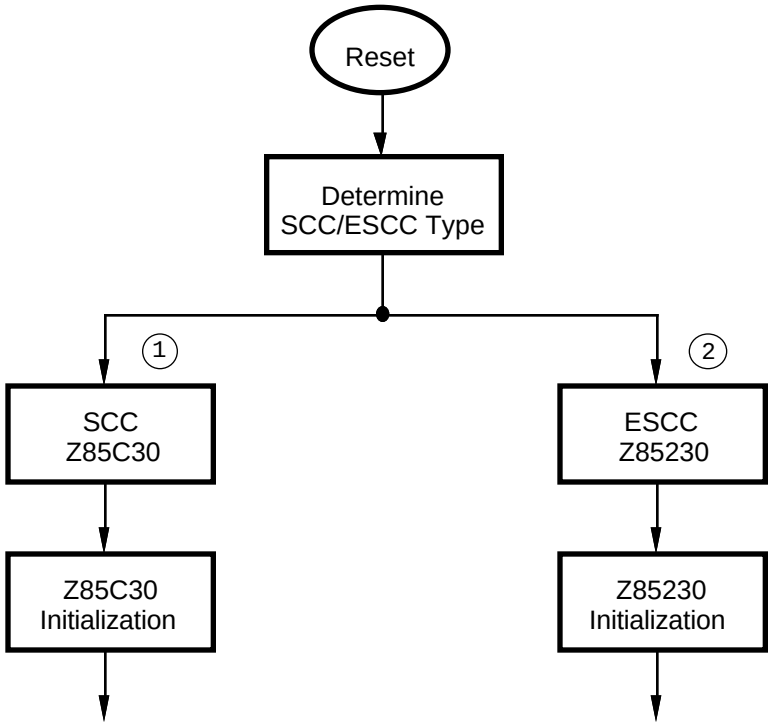



**Figure 2. Generic SCC/ESCC Drivers**

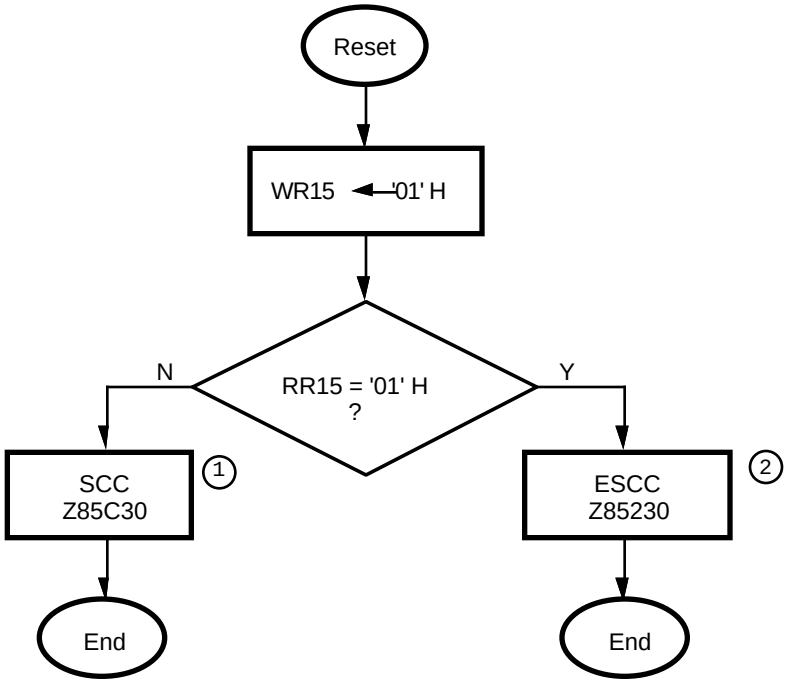
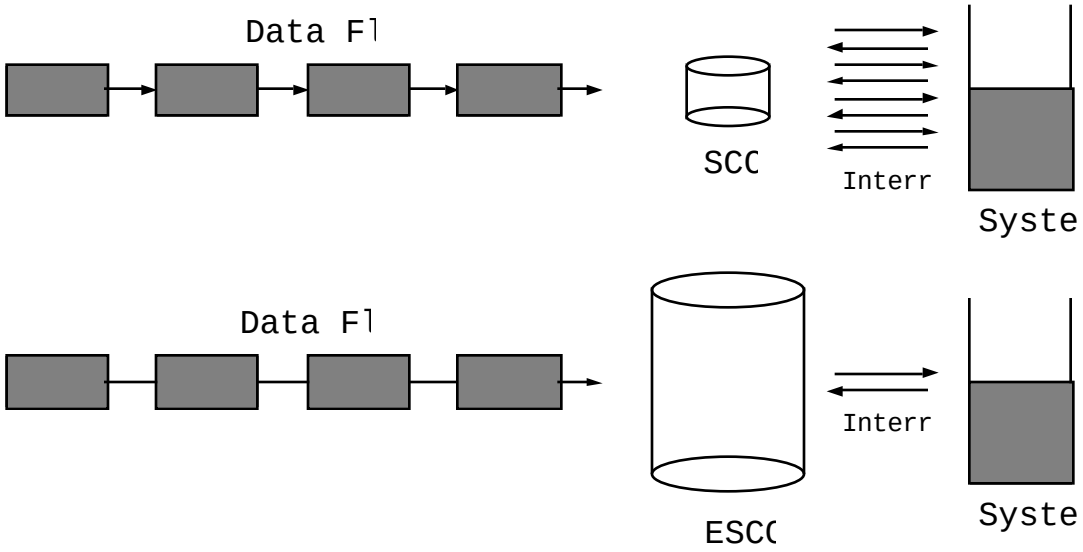



**Figure 3. SCC/ESCC Differentiation Flowchart**

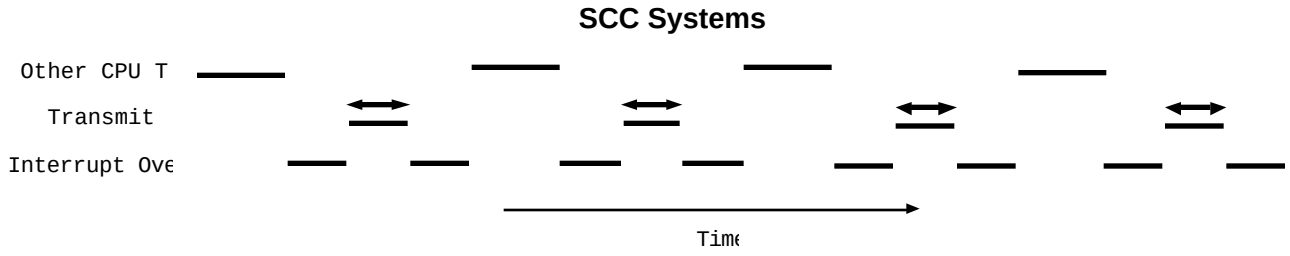
## ESCC SYSTEM BENEFITS

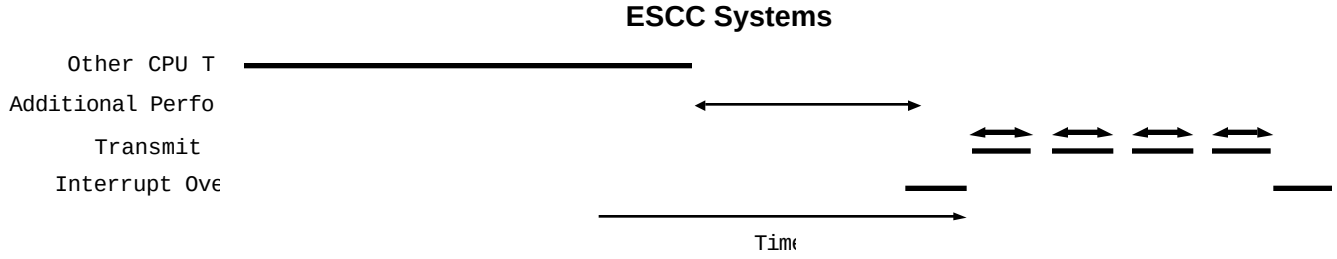
The Software Overhead sets the System Performance Limits. The ESCC's deeper FIFOs and other features significantly reduce the software overhead for each channel. This allows:

- More Channels Per System
- Faster Data Rates on Channels
- More CPU bandwidth available for other tasks
- Lower CPU Costs




**Interrupt Frequency Reduction**




**ESCC Reduces System Workload and Allows Extra Performance**

## TRANSMIT FIFO INTERRUPT

In the ESCC, transmit interrupt frequencies are reduced by a deeper Transmit FIFO and the revised transmit interrupt structure. If the WR7' D5 Transmit FIFO Interrupt Level bit is reset, the transmit interrupt is generated when the entry location of the FIFO is empty, i.e., more data can be written. This is downward compatible with a SCC Transmit Interrupt since the SCC only has a one-byte transmit buffer instead of a four-byte Transmit FIFO.

If WR7' D5 is set, the transmit buffer empty interrupt is generated when the transmit FIFO is completely empty. Enabling the transmit FIFO interrupt level, together with polling the Transmit Buffer Empty (TBE) bit in RR0, causes significant transmit interrupt frequency reduction. Transmit data is sent in blocks of four bytes (algorithm is illustrated in Figure 4). This helps to offload those systems which have long interrupt latency or a fully loaded Operating System.

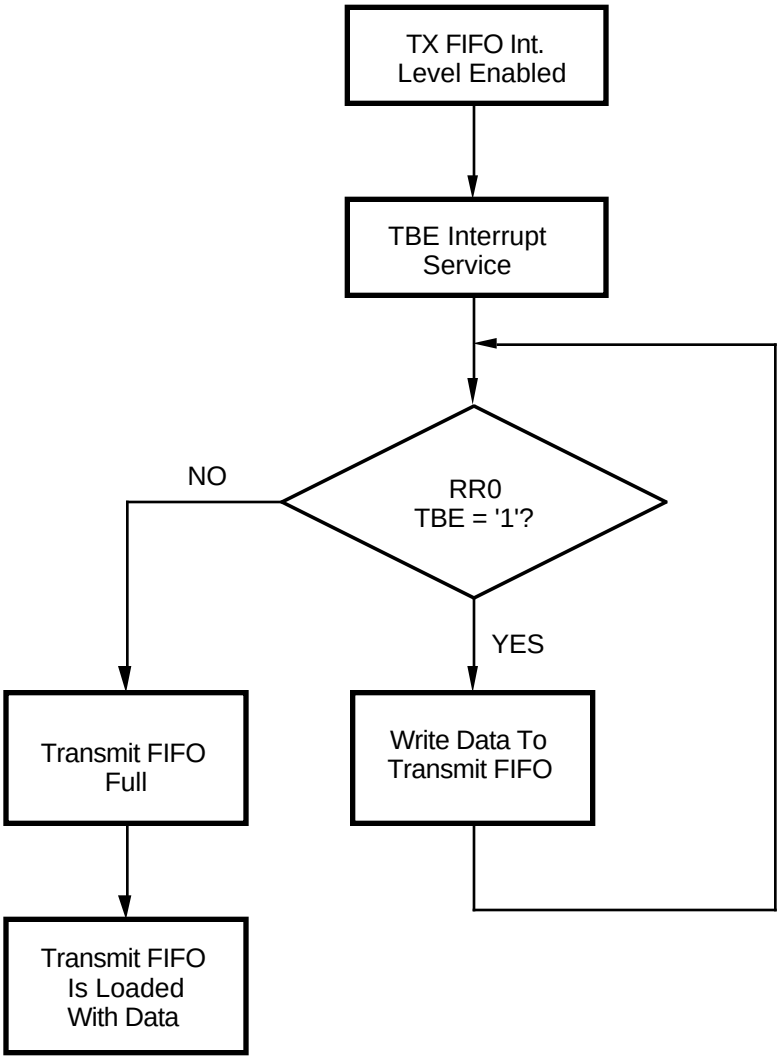



**Figure 4. Flowchart of Transmit Interrupt Service Routine to Reduce Transmit Interrupt Frequencies**

## RECEIVE FIFO INTERRUPT

In the ESCC, receive interrupt frequencies are reduced due to a deeper Receive FIFO and the revised receive interrupt structure.

If WR7' D3 Receive FIFO Interrupt Level bit is reset, the ESCC generates the receive character available interrupt on every received character. This is compatible with SCC Receive Character Available Interrupt. If WR7' D3 is set, the Receive Character Available Interrupt is triggered when the Receive FIFO is half full; the first four locations from the entry are still empty. By enabling the receive FIFO interrupt level, together with polling the Receive Character Available (RCA) bit in RR0, the receive interrupt frequencies are reduced significantly. Receive data is read in blocks of four bytes (Figure 5). This would help to offload systems which have a long interrupt latency and heavily loaded Operating Systems.

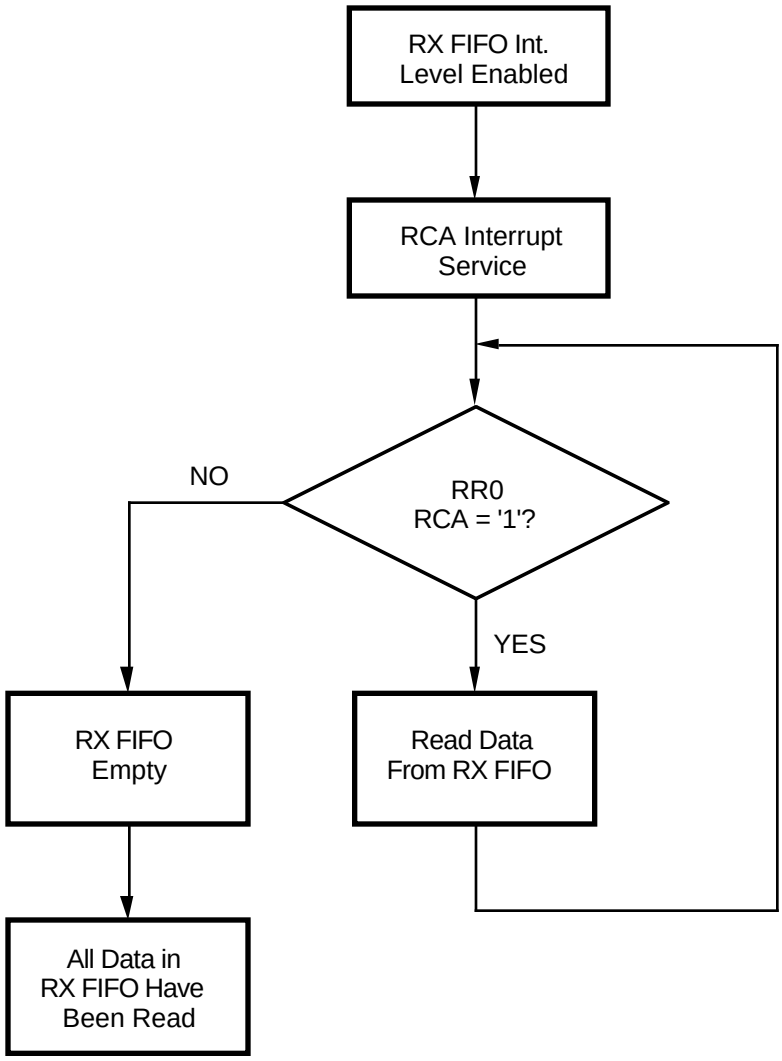



**Figure 5. Flowchart of Receive Interrupt Service Routine to Reduce Receive Interrupt Frequencies**

## AUTOMATIC /RTS DEASSERTION

Several SDLC enhancements are provided in the ESCC. The ESCC allows automatic /RTS deassertion at End Of Frame (EOF). The automatic /RTS deassertion is enabled by setting WR7' D2. If ESCC is programmed for SDLC mode and the Flag-On-Underrun bit (WR10 D2) is reset, with the RTS bit (WR5 D1) reset, /RTS is deasserted automatically at the last bit of the closing flag. It is triggered by the rising edge of the Transmit Clock (TxC - Figures 6 and 7).

/RTS is normally used in SDLC for switching the direction of line drivers. Automatic /RTS deassertion allows optimal line switching without any software intervention. The typical procedures are as follows:

1. Enable Automatic /RTS Deassertion
2. Before frame transmission, set RTS bit
3. Enable frame transmission
4. Reset RTS bit
5. RTS pin deassertion is delayed until the last rising TxC edge closing flag.

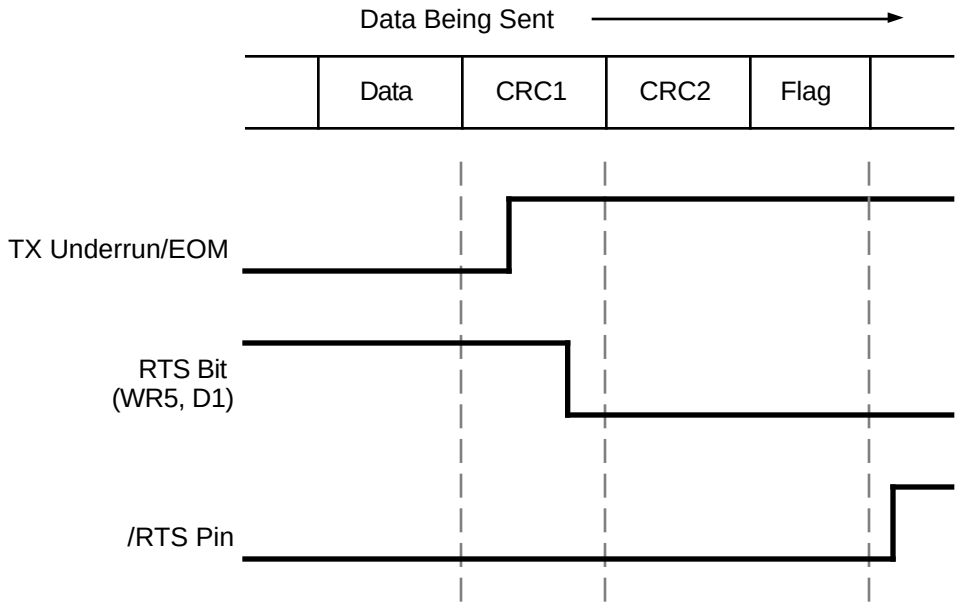



**Figure 6. /RTS Deassertion Timing**

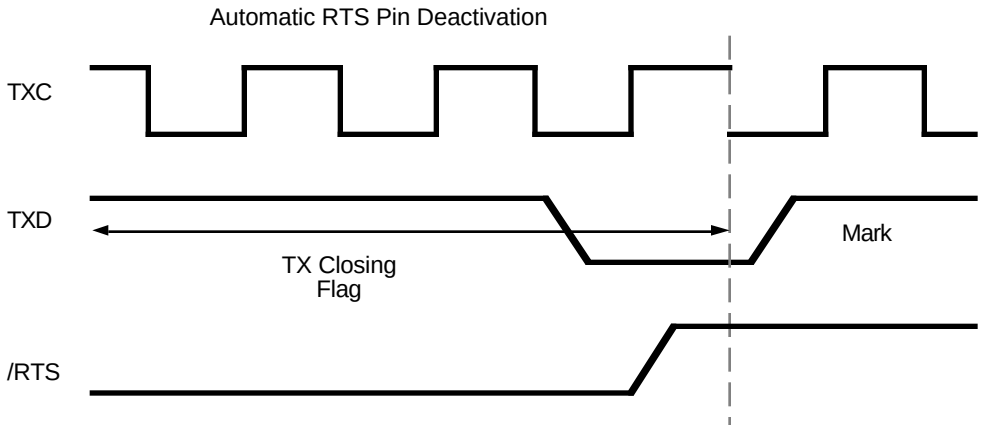



**Figure 7. /RTS Deassertion Sequence**

## AUTOMATIC OPENING FLAG TRANSMISSION

When Auto Tx Flag (WR7', D0) is enabled, the ESCC automatically transmits a SDLC opening flag before transmitting data. This removes:

1. Requirements to reset the mark idle bit (WR10 D3) before writing data to the transmitter, or;

2. Waiting for eight bit times to load the opening flag.

### TxD Forced High In SDLC With NRZI Encoding When Marking Idle After End Of Frame

When the ESCC is programmed for SDLC mode with NRZI encoding and mark idle (WR10 D6=0,D5=1,D3=1), TxD is automatically forced high when the transmitter goes to the mark idle state at EOF or when Abort is detected. This feature is used in combination with the automatic SDLC opening flag transmission to format the data packets between successive frames properly without any requirement in software intervention.

### Status FIFO Enhancement

ESCC SDLC Frame Status FIFO implementation has been improved to maximize ESCC ability to interface with a DMA-driven system (Technical Manual, 4.4.3). The Status FIFO and its relationship with RR1, RR6 and RR7 is shown in Figure 8.

Other special conditions (e.g., Overrun) generates special receive conditions and lock the Receiver FIFO (Figures 9 and 10).

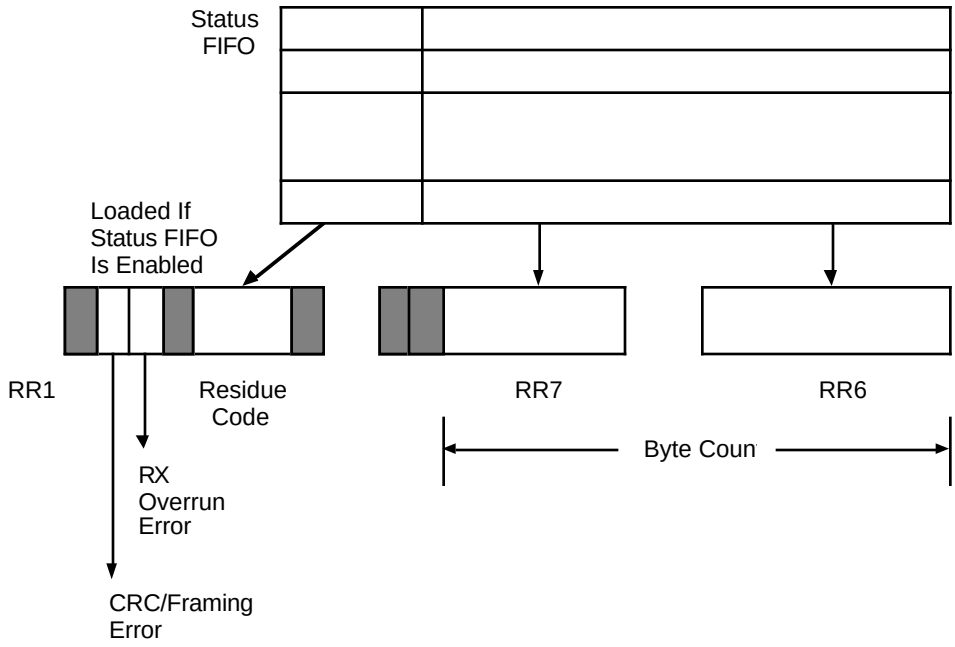



**Figure 8. Status FIFO**

SDLC Frame Status FIFO enhancement is enabled by setting WR15 D2. If it is enabled when EOF is detected, byte count and status from the Status FIFO are loaded into RR6, RR7 and RR1. This is used in DMA-driven systems. Historically, EOF is treated as a special condition. Special condition interrupts are triggered if any one of the below interrupts is enabled:

1. Receive Interrupt on First Character or Special Condition.
2. Interrupt on All Receive Characters or Special Conditions.
3. Special Receive Condition Only.

If 1 or 3 (above) is enabled, the data FIFO is locked after the interrupt is serviced by reading RR1 in the Status FIFO, as shown in Figure 11. This is commonly used in a DMA-driven system to avoid delivering useless information (e.g., EOF) to the data buffer. Locking the data FIFO is not desirable in systems with long interrupt latency and high data rate communications. The reason is the ERROR RESET command is necessary to unlock the FIFO. Data from the next frame may be lost if ERROR RESET fails to issue early.

This drawback is improved in the ESCC for a DMA driven system. By enabling interrupts on “Special Receive Conditions only” and SDLC status FIFO, EOF is treated differently from other special conditions. When EOF status reached the exit location of the FIFO:

1. A “Receive Data Available” interrupt is generated to signal that EOF has been reached.
2. Receive Data FIFO is not locked.

Because of these changes, the data from the next frame is securely loaded and the system processes the EOF interrupt. The only responsibility of the software is issuing the Reset Highest IUS before resuming normal operation (Figure 12).

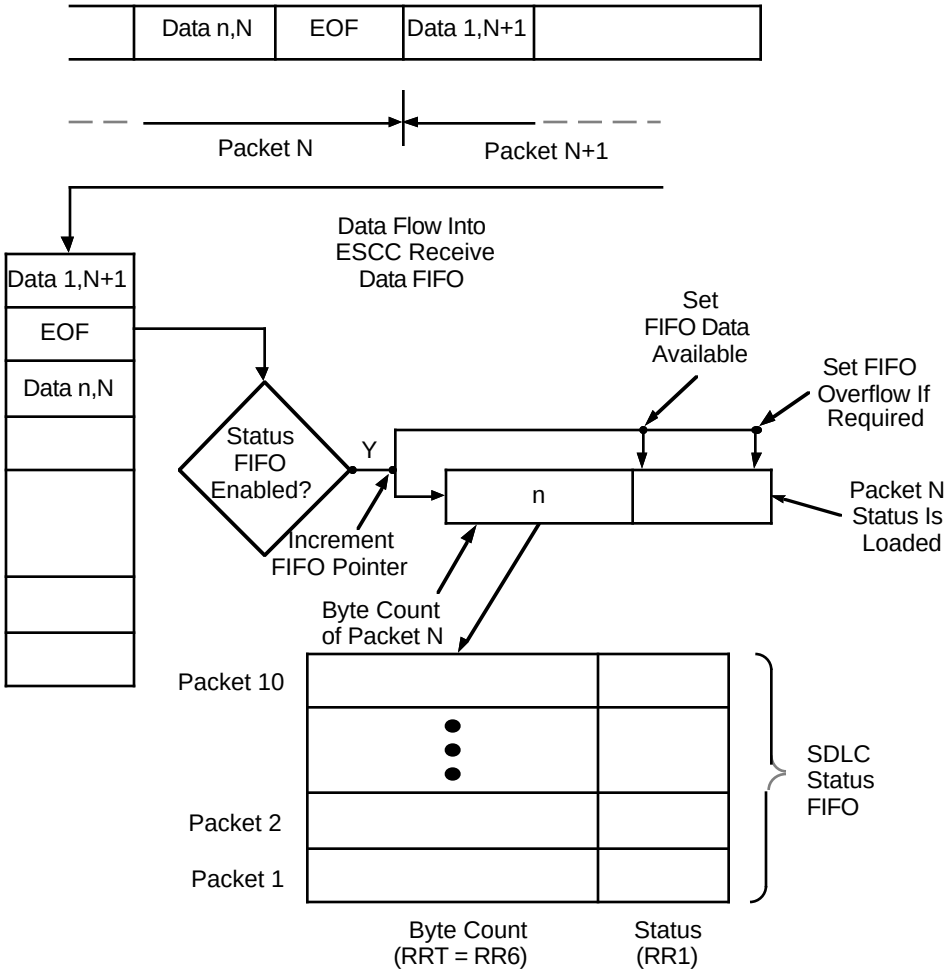
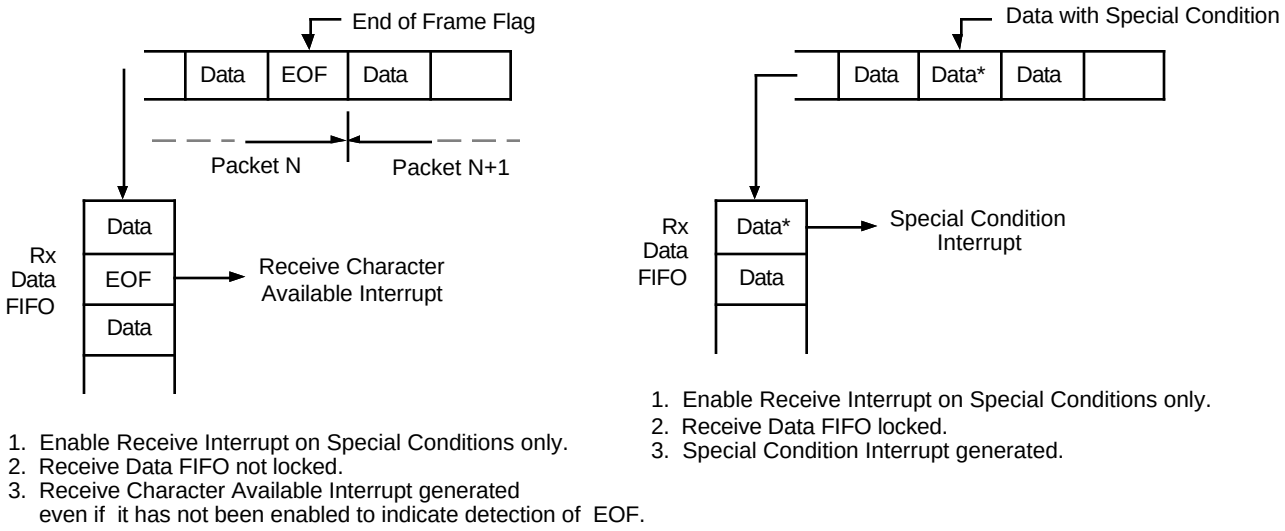



**Figure 9. Status FIFO Operation at End Of Frame**

## AUTOMATIC OPENING FLAG TRANSMISSION (Continued)




1. Enable Receive Interrupt on Special Conditions only.
2. Receive Data FIFO not locked.
3. Receive Character Available Interrupt generated even if it has not been enabled to indicate detection of EOF.

1. Enable Receive Interrupt on Special Conditions only.
2. Receive Data FIFO locked.
3. Special Condition Interrupt generated.

**Figure 10. SDLC Status FIFO Anti-Lock**

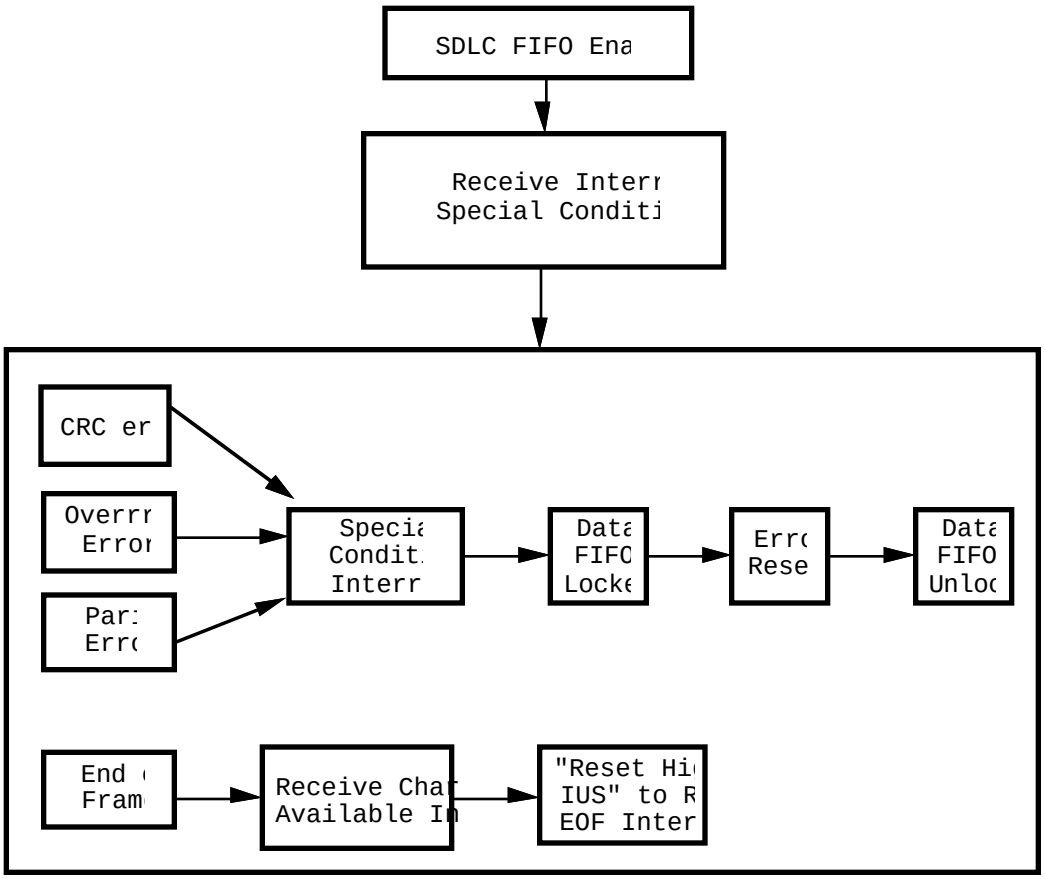



**Figure 11. Receive Interrupt Mechanism 1**

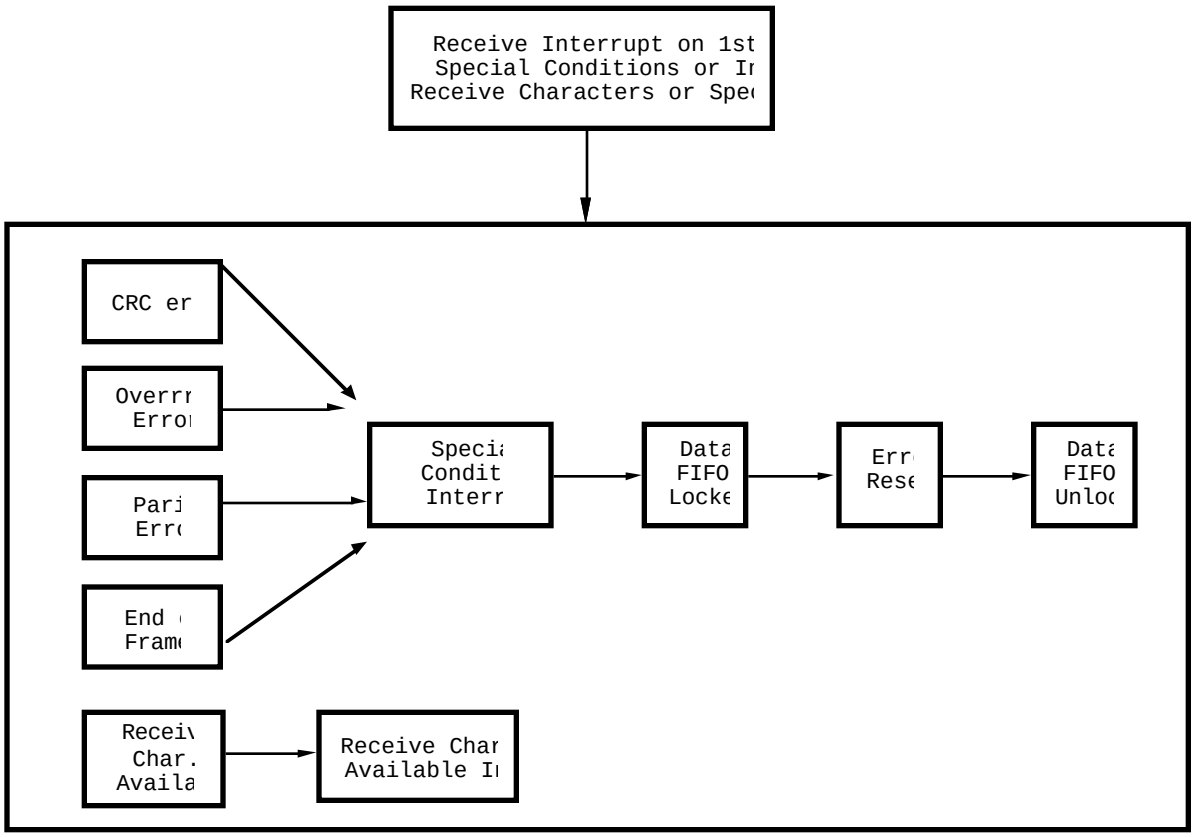



**Figure 12. Receive Interrupt Mechanism 2**

## DMA Request on Transmit Deactivation Timing /DTR//REQ.

Timing implementation in the ESCC has been improved to make it more compatible with the DMA cycle timing (Reference Tech Manual, Section 2.5.2; DMA Request on Transmit).

## Transmission of Back-To-Back Frames with a Shared Flag.

The ESCC provides facilities to allow transmission of back-to-back frames with a shared flag between frames (Figure 13).

In the ESCC, if the Automatic End Of Message (EOM) Reset feature is enabled (WR7' D1=1), data for a second frame is written to the transmit FIFO when Tx Underrun/EOM interrupt has occurred. This allows application software sufficient time to write the data to the transmit FIFO while allowing the current frame to be concluded with CRC and flag.

In the SCC, Transmission of Back-to-Back Frames is more difficult because (Figure 14):

1. Data cannot be written to the transmitter at EOF until a Transmit Buffer Empty interrupt is generated after CRC has completed transmission.

2. Automatic EOM Reset is not available in the SCC. Application software has to issue the “Reset Tx/Underrun EOM” command manually. The software overhead limits the next frame data to deliver immediately after the preceding frame has been concluded with CRC and Flag.

## AUTOMATIC OPENING FLAG TRANSMISSION (Continued)

Requirements: Automatic EOM Reset and Automatic Opening Flag features are enabled.

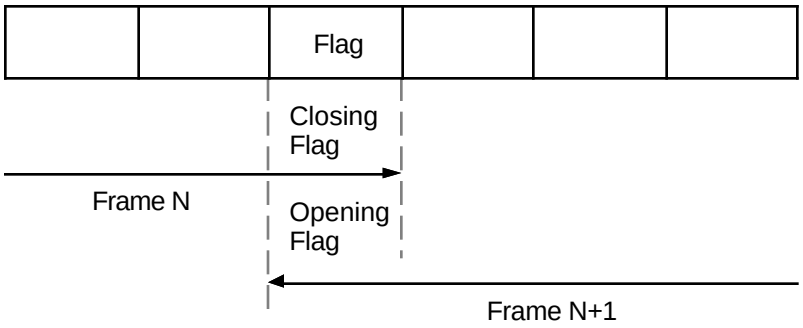



**Figure 13. Transmission of Back-to-Back Frames with a Shared Flag**

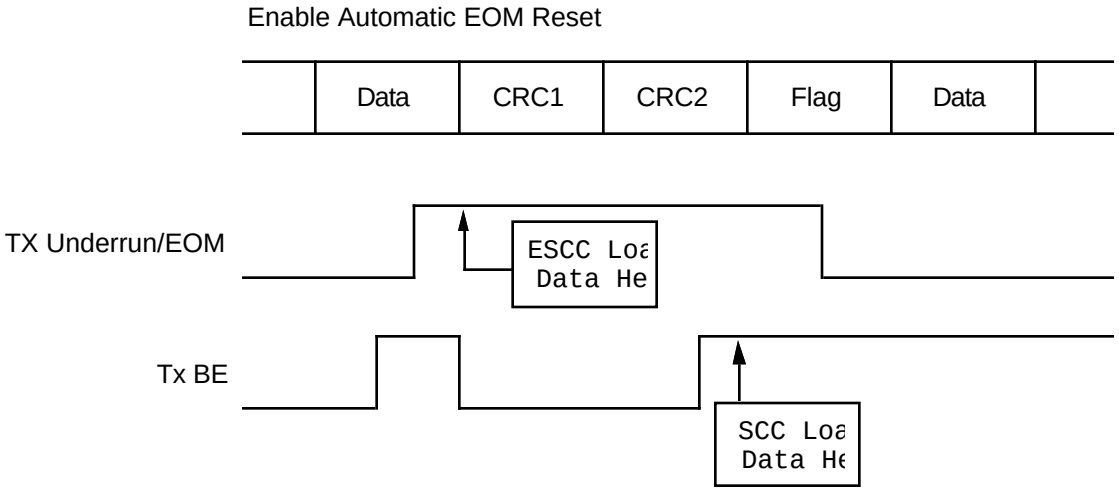
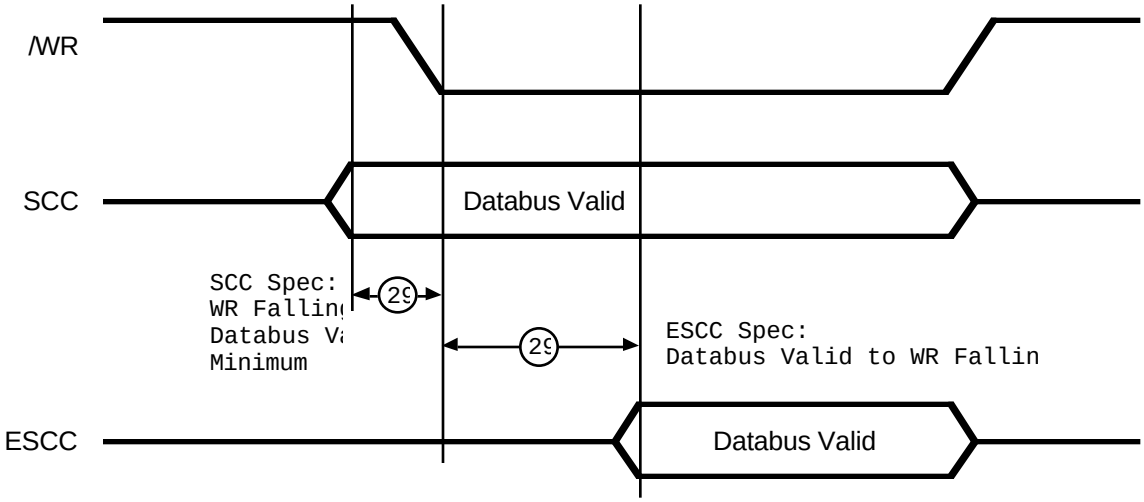



**Figure 14. Operation of Shared Flag Transmission**

## MODIFIED WRITE TIMING

In the SCC write cycle, the SCC assumes the data is valid when /WR is asserted (Figure 15). This assumption is not valid for some CPUs, e.g., the Intel 80X86. The /WR signal from this CPU needs to delay for one more clock to initiate the write cycle. Additional hardware is required.

In the ESCC, write cycle timing has been modified so that data becomes valid a short time after write (approx. 20 ns). Therefore, if the data pins from the Intel CPU are connected directly to the ESCC, no additional logic is required.




Databus latched after falling edge of WR saves external logic required to delay WR until databus is valid. Typically needed with Intel CPUs.

**Figure 15. Modified Write Timing**

# TECHNICAL CONSIDERATIONS WHEN IMPLEMENTING LOCALTALK LINK ACCESS PROTOCOL

*The LLAP Protocol is an important part of the Appletalk network system. It manages access to the node-to-node transmission of network data packets, governs access to the link, and provides a means for nodes to discover valid addresses...all error free.*

## INTRODUCTION

The LLAP (LocalTalk Link Access Protocol) is the ISO/OSI (International Standards Organization/Open Systems Interconnection) link layer protocol of the AppleTalk network system. This protocol manages the node-to-node transmission of data packets in the network. LLAP governs access to the link and provides a means for nodes to discover valid addresses. It does not guarantee packet delivery; it does guarantee that those packets that are delivered are error-free.

This Appnote (Application Note) does not address the architectural issues of writing a driver but it does focus on the details of using an SCC to send and receive LLAP frames. However, some of the problems of transmitting and receiving LLAP frames are discussed, using sample code written for Zilog's Z80181 Emulation Adapter Board. Also, the problems of sending sync pulses, timing transmissions and determining that a frame has been received properly will be discussed.

## GENERAL DESCRIPTION

The LocalTalk Link Access Protocol (LLAP) is the ISO-OSI link layer protocol of the AppleTalk network system using LocalTalk. Along with ELAP (the corresponding Ethernet link layer protocol) and TLAP (the Token Ring link layer protocol), it provides the foundations upon which the other protocols rest. The LLAP protocol supports the node-to-node transmission of packets used by DDP and RTMP to route packets around the internetwork; DDP, in turn, supports the name binding functions of NBP, the reliable frame delivery of ATP, and the rest of the AppleTalk protocol stack.

A majority of the difficult timing and all of the hardware interface problems crop up in the LLAP driver. These problems are so difficult that it makes sense to start writing such a driver by writing experimental routines that transmit and receive frames. This App Note addresses the intricacies of the interframe and interdialog timings before trying to engineer code that will truly be a driver. Also, some of the experimental routines to run on the Z80181 Emulation Adapter Board will be explained.

The LLAP provides the basic transmission of packets from one node to another on the same network. LLAP accepts packets of data from clients residing on a particular node and encapsulates that data into its proper LLAP data packet. The encapsulation includes source and destination addresses for proper delivery. LLAP ensures that any damaged packet is discarded and rejected by the destination node. The LLAP makes no effort to deliver damaged packets.

### Carrier Sense Multiple Access with Collision Avoidance

It is LLAP's responsibility to provide proper link access management to ensure fair access to the link by all nodes on that network. The access discipline that governs this is known as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). A node wishing to gain access to the link must first sense that the link is not in use by any other node (carrier sense); if the link has activity, then the node wishing to transmit must defer transmission. The ability of LLAP to allow multiple access to the link also

## GENERAL DESCRIPTION (Continued)

leaves room for possible collisions with other data packets. LLAP attempts to minimize this probability (collision avoidance).

Two techniques are used by LLAP in its implementations of CSMA/CA. LLAP outlines this procedure but falls short in endorsing which hardware to use. (The SCC is, of course, used by Apple.) The first technique takes advantage of the distinctive 01111110 flag bytes that encapsulate the data packet (note that this implies that SDLC is used). LLAP stipulates that a minimum of two flags precede each of these data packets. The leading flag characters provide byte synchronization and give a clue to any listener that some other node is using the link at a particular time (use the Hunt bit in RR0 if the SCC is used).

In SDLC mode, the receiver automatically synchronizes on the flag byte and resets the Hunt bit to zero. The SCC has some latency in detecting these flag bytes due to the shifter, etc. This is not ideal because the node needing to transmit may determine that the link is free, when in fact the flag bytes are still being shifted into its receiver (i.e., the link is not idle at all).

A closing flag is also needed to fully encapsulate the data packet. LLAP requires that 12 to 18 ones be sent after this closing flag. The LocalTalk hardware (i.e., the SCC) interprets this as an abort sequence and causes the node's hardware to lose byte sync; this then confirms that the current sender's transmission is over. In SDLC mode, seven or more contiguous 1's in the receive data stream forces the receiver into Hunt (Hunt bit set) and an External/Status interrupt can be generated. This is important because the node wishing to use the bus can simply wait for this interrupt before preparing to transmit it's packet.

LLAP uses a second technique in its carrier sensing. LLAP requires that a synchronization pulse for an idle period of at least two bit times be transmitted prior to sending the RTS handshaking frame (Figure 1). This synchronization is obtained by first enabling the hardware line so that an edge is detected by all the receivers on the network. This initial edge is perceived as the beginning of the clocking period. It is soon followed by an idle period (a period with no carrier) of at least two bit times. All the receivers on the network see this idle period and assume that the clock has been lost (missing clock bit set on RR10 ). This method is much more immediate than the byte flag synchronization method and provides a quicker way of determining whether the link is in use. Unfortunately, an interrupt is not generated by this missing clock and, therefore, polling must be implemented.

The Z80181 code used for polling the missing clock bit is approximately fifty clock cycles which at 10 MHz is about 5 μsec or about one bit time. This is still relatively quicker than the time required for the SCC to reset the Hunt bit (the flag character takes at least eight bit times for it to be shifted through the buffer before the Hunt bit is reset to zero). Synchronization pulses can be sent before every frame but because of the time constraints associated with the interframe gaps it makes sense to send such pulses only before the lapENQ and lapRTS frames.

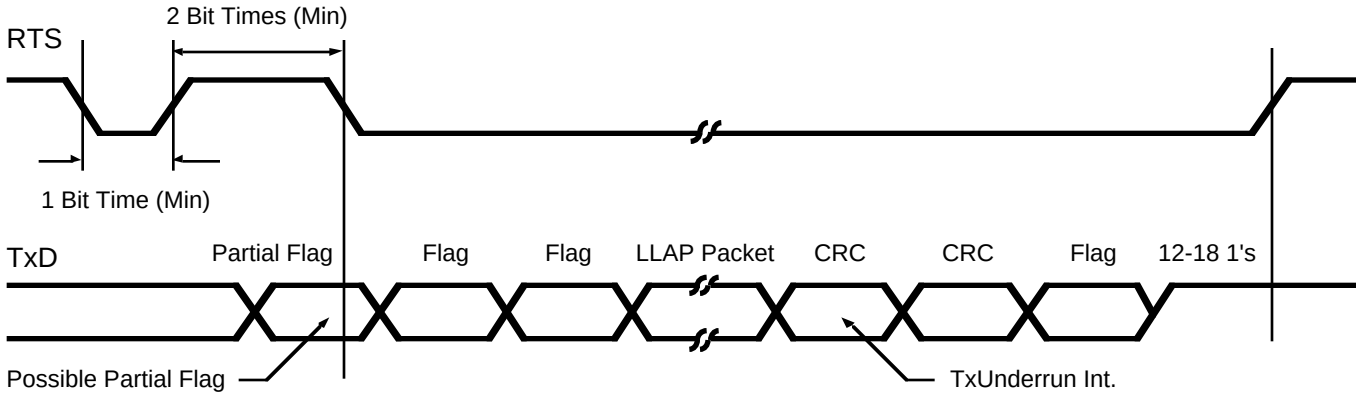



**Figure 1. CSMA/CA Synchronization Pulse Timing Diagram**

## Dynamic Node ID

LLAP requires the use of an 8-bit node identifier number (node ID) for each node on the link. Apple had decided that all LLAP nodes must have a dynamically assigned node ID. A node would assign itself its unique address upon activation. It is then up to that particular node to ascertain that the address it had chosen is unique. A node randomly chooses an 8-bit address (for example, the refresh register value on the Z80181 is added to a randomly chosen value on the receive buffer to obtain a pseudo random 8-bit address).

The node then sends out an LLAP Enquiry control packet to all the other nodes and waits for the prescribed interframe gap of 200 µsec. If another node is already using this node ID, then that node must respond within 200 µsec with a LLAP Acknowledgment control packet. The new node must then rebroadcast a new guess for its node ID. If a LLAP Acknowledgment packet is not received within 200 µsec then the new node assumes that the address is indeed unique. The new node must rebroadcast the LLAP enquiry packet several more times to account for cases when the packet could have been lost or when the guessed node ID is busy and could have missed the Enquiry packet.

## LLAP Packet

LLAP packets are made up of three header bytes (destination ID, source ID and LLAP type) and 0 to 600 bytes of variable length data. The LLAP type indicates the type of packet that is being sent. 80H to FFH are reserved as LLAP control packets. The four LLAP control packets that are currently being used are: The lapENQ, which is used as enquiry packet for dynamic node assignments; the lapACK, which is the acknowledgment to the lapENQ; the lapRTS, which is the request to send packet that notifies the destination of a pending transmission; and the lapCTS, which is the clear-to-send packet in response to the RTS packet. Control packets do not contain data fields.

## LLAP Packet Transmission

LLAP distinguishes between two types of transmissions: a directed packet is sent from the source node to a specific destination node through a directed transmission dialog; a broadcast packet is sent from the source node to all nodes on the link (destination ID is FFH) through a broadcast transmission dialog. All dialogs must be separated by a minimum Inter Dialog Gap (IDG) of 400 µsec. Frames within these dialogs must be separated from each other with a maximum Inter Frame Gap (IFG) of 200 µsec.

The source node uses the physical layer to detect the presence or the absence of data packets on the link. The node will wait until the line is no longer busy before attempting to send its packets. If the node senses that the line is indeed busy, then this node must defer. When the node senses that the line is idle, then the node waits the minimum IDG plus some randomly generated time before sending the packet (the line must remain idle throughout this period before attempting to send the packet). The initial packets to be sent are handshaking packets. The first packet sent by the source node to its destination node is the RTS packet. The receiver of this RTS packet must return a CTS packet within the allowable maximum IFG. The source node then starts transmitting the rest of its data packet upon receiving this CTS.

Collisions are more likely to occur during the handshaking phase of the dialog. The randomly generated time that is added to the IDG tends to help spread out the use of the link among all the transmitters. A successful RTS to CTS handshake signifies that a collision did not take place. An RTS packet that collides with another frame has corrupt data that shows up as a CRC error on the receiving or the destination node. Upon receiving this, the destination node infers that a collision must have taken place and abstains from sending its CTS packet. The source or the transmitting node sees that the CTS packet was not received during the IFG and also infers that a collision did take place. The sending node then backs off and retries.

The LLAP keeps two history bytes that log the number of deferrals and collisions during a dialog. These history bytes help determine the randomly generated time that is added to the IDG. The randomly generated time is readjusted according to the traffic conditions that are present on the link. If collisions or deferrals have just occurred on the most recently sent packets, then it can be assumed that the link has heavier than usual traffic. Here, the randomly generated number should be a larger number in order to help spread out the transmission attempts. Similarly, if the traffic is not so great, then the randomly generated number should be smaller, thus reducing the dispersion of the transmission attempts.

## LocalTalk Physical Layer

LocalTalk uses the SDLC format and the FM0 bit encoding technique. The RS-422 signalling standard for transmission and reception was chosen over the RS-232 because a higher data rate over a longer physical distance is required. LocalTalk requires signals at 230.4 Kbits per second over a distance of 300 meters.

## HARDWARE CONFIGURATION

As shown in Figure 2, the hardware used to implement this LLAP driver consists of the Z80181 (an integration of the Z80180 compatible MPU core with one channel of a Z85C30 SCC, Z80 CTC, two 8-bit general-purpose parallel ports and two chip select signals) operating at 10 MHz, a 3.6864 MHz clock source and an RS-422 line driver with tri-state.

The SCC's clocking scheme decouples the micro-processor's clock from the communication clock (Figure 3). The DPLL uses the /RTxC pin as its source. The /RTxC also drives the Baud Rate Generator which divides its input by sixteen. The resulting 230.4 kHz signal is then used as transmitter clock. This 230.4 kHz signal is also used by one of the Z80181's counter/timer trigger inputs (Z80 CTC's channel 1) which is used to count the number of elapsed bit times. In counter mode, each active edge to the CTC's CLK/TRG1 input causes the downcounter of the CTC to be decremented. The /TRxC pin is programmed as BRG output and is connected to the CLK/TRG1 input through an external wire.

The /RTS signal is used to tri-state RS-422 to allow other node transmitters to drive the line. This signal is asserted and deasserted through bit1 of the SCC's Write Register 5.

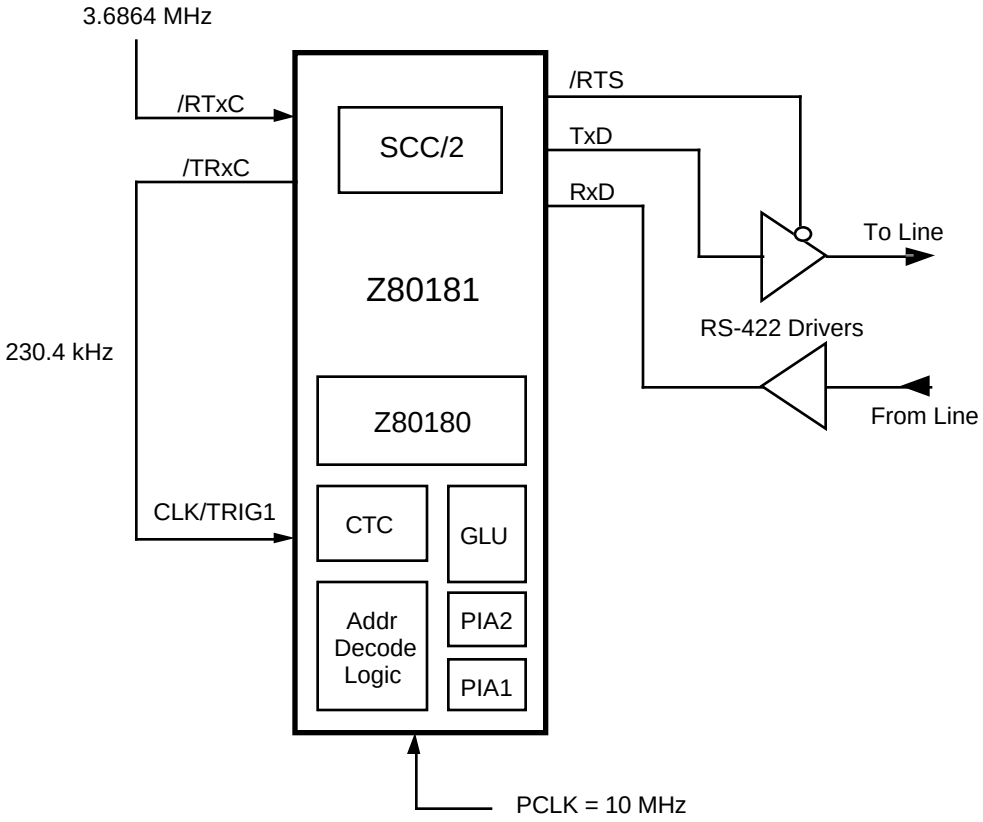



**Figure 2. Driver Hardware Configuration**

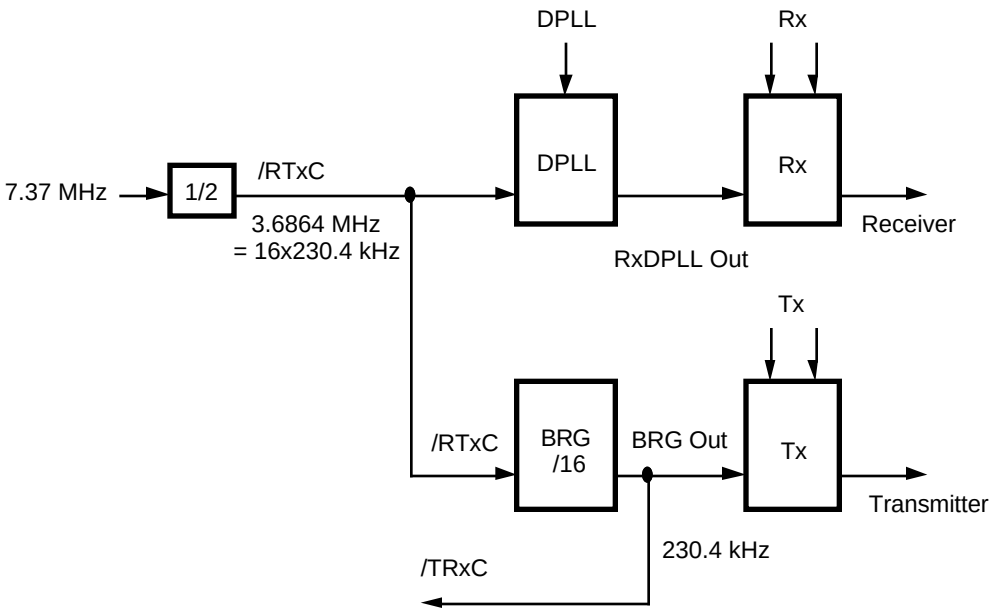



**Figure 3. SCC Clocking Scheme**

Listing 1 (Reference Appendix A for Listings 1 through 4) shows the assembler code for this SCC initialization. Note that the SCC is treated as a peripheral by the Z80181's MPU. For example, an I/O write to the scc_cont (address e8H) or to the scc_data (address e9H) is a write to the SCC's control and data registers, respectively. As shown in Listing 1, the SCC is initialized by issuing I/O writes to the pointer and then to the control registers in an alternating fashion. It is therefore very important that all interrupts are disabled during this initialization routine.

The SCC is initially reset through software before proceeding to program the other write registers. A NOP is sufficient to provide the four PCLKs required by the SCC recovery time after a soft reset. The SCC is programmed for SDLC mode. The receive, transmit and external interrupts are all initially disabled during this initialization. Each of these interrupt sources are enabled at their proper times in the main program. The SCC is programmed to include status information in the vector that it places on the bus in response to an interrupt acknowledge cycle (see Listing 4 of the SCC interrupt vector table for all the possible sources).

Since SDLC is bit-oriented, the transmitter and receiver are both programmed for 8 bits per character as required by LLAP. Address filtering is implemented by setting the Address Search Mode bit 2 on WR3. Setting this bit causes messages with addresses not matching the address programmed in WR6 and not matching the broadcast address to be rejected. Values in WR10 presets the CRCs to ones, sets the encoding to FM0 mode and makes certain that transmission of flags occur during idle and underrun conditions. WR11 is set so that the receive clock is sourced by the DPLL output; the transmit clock is sourced by the Baud Rate Generator output; /TRxC's output is from the BRG. The input to the BRG is from the /RTxC.

The BRG's time constant is loaded in WR13 and WR12 so that the /RTxC's 3.6864 MHz signal is divided by 16 in order to obtain a 230.4 kHz signal for the transmitter clock. WR14 makes certain that the DPLL is disabled before choosing the clock source and operating mode. The DPLL is enabled by issuing the Enter Search Mode in WR14.

## TRANSMITTING A LLAP FRAME

Listing 2 shows the assembler code for subroutine txenq, which sends an lapENQ frame on the line once the system has determined that the line is quiet. Note that this routine can easily be generalized to send any frame.

The first responsibility of txenq is to send the sync pulse required by the CSMA/CA protocol. To do this, txenq sets the /RTS pin active low, enabling the transmitter drivers, and then sets it high again to disable them. In order to satisfy the requirements of the CSMA/CA protocol, the transmitter drivers must remain off for at least one bit time (4.3 μsec) to guarantee that all the receivers see at least one transition. Our routine satisfies this requirement because the two ld instructions (7 T states each), the two nop instructions (4 T states each) and the two "out" instructions (11 T states each) required to set the /RTS line high, take more than 4.3 μsec to execute on the 10 MHz Z80181. The transmitter drivers must then remain off for at least two bit times in order to ensure that all receivers lose clock; again, the routine depends upon the time required to execute the instructions before we turn the transmitter drivers on again.

After sending the sync pulse and waiting the required period of silence, txenq turns on the transmitter drivers to send the frame. Now, the routine must wait while the SCC sends out the leading flags. This takes 16 bit times, and since the SCC does not tell when this has happened, the transmit routine has no choice but to time this. Our routine does this by calling bit time, which is discussed below.

When the two flags have been transmitted, the first data byte is written to the data register of the SCC. Thereafter, the routine polls the SCC status register, and when that register shows the transmit buffer register is empty, the routine sends the next data character. This polling method can obviously be replaced by an interrupt routine that is entered when the transmit buffer is empty or by setting up the Z80181's DMA to send characters on demand to the SCC.

After the first data byte is transmitted, the txenq routine sets the SCC to mark on idle so that the abort is sent when the frame is over. This operation can be done any time after the first data character has been placed in the transmit buffer and before the trailing flag is shifted out. Txenq asserts this mark on idle command after the first character is placed in the transmit buffer so that LLAP has control and that no issues of latency may arise (particularly in designs using interrupt or DMA).

After the last data byte is written to the SCC, the transmit routine must wait while the last data byte (the one that the SCC had just sent to shifter), the two CRC bytes, one flag byte and 12 to 18 bit times of marking are transmitted. This total of 44 to 50 bit times is again timed by bittime. When bittime indicates that enough time has elapsed, the transmitter drivers are turned off.

Since our hardware includes connecting the output of the baud rate generator to the input of counter/timer 1 on the Z80181, that counter timer counts the bit times. The bit time routine feeds an appropriate count value into the counter and enables an interrupt routine to receive control when the count expires. The interrupt routine ctc1int, shown in Listing 4, sets the timeflag which the transmit routine polls.

Note that the call to bittime, the interrupt routine, the polling code and the length of time it takes to write to the SCC registers after a polling loop is exited, all take up a time that can be a significant number of bits. In order to do these timings accurately, calculate the number of PCLK cycles required to execute these pieces of code and to adjust the counter value that bittime requires. This adjustment is dependent on the hardware configuration and on the exact implementation details of the code.

Note, incidentally, that software must put the entire frame into the transmit register, including the addresses. The SCC does not generate addresses even when set in WR6.

## RECEIVING LLAP FRAMES

In the experiments, the interrupt routines were used to receive characters and to handle special conditions when the frame is complete. Listing 3 shows the interrupt handlers that are entered when the SCC receives a character and when the SCC interrupts for a special condition (typically, end of frame). As with transmission, it is obvious that we could receive characters by polling the SCC (using up all available CPU cycles) or by using DMA (using up very few). It is estimated that the recint routine uses up about 1/3 of the available 34 μsec (4.3 μsec x 8-bit times) cycles on a 10 MHz processor.

The recint routine moves each character as it is received from the SCC to a memory buffer and increments the buffer pointer. The frame's data length is checked to make certain that the maximum allowable frame size is not exceeded.

The spcond interrupt handler checks the status of the SCC to find out what has happened. The presence on an overrun condition or a CRC error is flagged as a receive frame error.

The second routine decrements the receiver buffer address by two to account for the two CRC bytes that are read from the SCC before the special condition interrupt occurs. Note that the SCC does not filter these CRC bytes, nor does it filter the address byte. Everything received after the leading flags and before the trailing flags appears in the receive buffer.

One difficulty that arises in LLAP that was not addressed here is that the receipt of a frame very often creates an obligation to send a frame back to the sender within the interframe gap, which is 200 µsecs. This difficulty is even greater than it might appear. The 200 µsec gap starts when the frame is received; it ends when the leading flags and destination address of the response are sent. Start sending the response soon enough to allow sending two leading flags (plus a possible leading flag fragment) and the first data character, and to allow for the 3-bit delay in the SCC shifter. Therefore, start sending early enough to transmit 35 bits before the interframe gap expires, or about 70 µsecs after you receive the frame.

## CONCLUSIONS

The problems of sending the sync pulses, the timing of transmission packets, and the problems associated with the reception of packets as defined by LLAP are handled by the Z80181 and its peripherals. It was demonstrated that LLAP frames can be transmitted and received by using the straight forward polling method and by using interrupt routines. In a much busier environment where the processor cannot strictly be an LLAP engine, other methods such as using DMA in a fully interrupt driven environment must be used. It was also demonstrated that severe CPU overhead is used in setting up the sync pulses, timing out delays, etc., before each LLAP frame. A modified SCC that transmits and receives special LLAP frames helps in off loading some of this overhead, hence freeing the CPU to do other tasks.

## APPENDIX A

## Listing 1- Asembler Code for SCC Initialization

**LISTING 1**

```
                        475                                 ;*************************************
                        476                                 ;subroutine to initialize scc registers
                        477                                 ;*************************************
000001e2                478 initscc:
000001e2 f3             479         di                      ;disable int while programming scc
000001e3 f5             480         push    af
000001e4 c5             481         push    bc
000001e5 e5             482         push    hl
                        483
000001e6 3e09           484         ld      a,09h           ;WR9
000001e8 d3e8           485         out     (scc_cont),a    ;point to scc register
000001ea 3e80           486         ld      a,80h           ;channel  reset
000001ec d3e8           487         out     (scc_cont),a    ;scc register value
000001ee 00             488         nop                     ;delay needed after scc reset
                        489
                        490
000001ef 21Wwww         491         ld      hl,scctable     ;fetch start of scc init table
000001f2                492 scc1:
000001f2 7e             493         ld      a,(hl)          ;fetch register pointer value
000001f3 feff           494         cp      0ffh
000001f5 caWwww         495         jp      z,finscc        ;if reg a =0ffh then initscc finished
000001f8 d3e8           496         out     (scc_cont),a
000001fa 23             497         inc     hl
000001fb 7e             498         ld      a,(hl)
000001fc d3e8           499         out     (scc_cont),a
000001fe 23             500         inc     hl
000001ff c3R000+01f2,   501         jp      scc1            ;loop back
                        502
00000202                503 scctable:
00000202 04             504         db      04h             ;WR4
00000203 20             505         db      00100000b       ;sdlc uses 1x,sdlc mode,no parity
                        506
00000204 01             507         db      01h             ;WR1
00000205 00             508         db      00h             ;nothing,rx,tx and ext int disabled
                        509
00000206 02             510         db      02h             ;WR2
00000207 00             511         db      00h             ;vector base is 00h
                        512
00000208 03             513         db      03h             ;WR3
00000209 cc             514         db      0cch            ;rx 8b/char,rx crc enabled,address
                        515                                 ;search mode for adlc address filtering
                        516                                 ;rx disabled.
                        517
0000020a 05             518         db      05h             ;WR5
0000020b 60             519         db      60h             ;tx 8b/char, set rts to disable drivers
                        520
0000020c 06             521         db      06h             ;WR6
0000020d 00             522         db      00h             ;address field='myaddress' in main pgm
                        523
0000020e 07             524         db      07h             ;WR7
0000020f 7e             525         db      7eh             ;flag pattern
                        526
00000210 09             527         db      09h             ;WR9
00000211 01             528         db      01h             ;stat low, vis therefore vector returned
                        529                                 ;is a variable depending on the source
                        530                                 ;of the interrupt.
                        531
```

```
00000212 0a     532             db      0ah             ;WR10
00000213 e0     533             db      0e0h            ;crc preset to one,fm0, flag idle/undr
                534
00000214 0b     535             db      0bh             ;WR11
00000215 f6     536             db      0f6h            ;rtxc=xtal,rxc=dpll,txc=brg,trxc=brg out
                537
00000216 0c     538             db      0ch             ;WR12
00000217 06     539             db      06h             ;brg tc low,for 230.4kbps using rtxc=3.68MHz
                540
00000218 0d     541             db      0dh             ;WR13
00000219 00     542             db      00h             ;brg tc high
                543
0000021a 0e     544             db      0eh             ;WR14
0000021b 60     545             db      60h             ;disable dpll
                546                                     ;no local loop back,brg source=rtxc
                547
0000021c 0e     548             db      0eh             ;WR14
0000021d c0     549             db      0c0h            ;select fm mode
                550                                     ;no local loop back,brg source=rtxc
                551
0000021e 0e     552             db      0eh             ;WR14
0000021f a0     553             db      0a0h            ;dpll source=rtxc,
                554                                     ;no local loop back,brg source=rtxc
                555
00000220 0e     556             db      0eh             ;WR14
00000221 20     557             db      20h             ;enter search mode
                558                                     ;no local loopback
                559
00000222 0e     560             db      0eh             ;WR14
00000223 01     561             db      01h             ;null,no local loopback,enable the brg
                562
00000224 03     563             db      03h             ;WR3
00000225 cc     564             db      0cch            ;rx 8b/c,enable rx crc,addrs src,rx disable
                565
00000226 0f     566             db      0fh             ;WR15
00000227 00     567             db      00h             ;ext/stat not used
                568
                569                                     ;WR0
00000228 10     570             db      10h             ;reset ext/stat once
00000229 10     571             db      10h             ;reset ext/stat twice
                572
0000022a 01     573             db      01h             ;WR1
0000022b 00     574             db      00h             ;disable all int sources
                575
0000022c 09     576             db      09h             ;WR9
0000022d 09     577             db      09h             ;enable int
0000022e ff     578             db      0ffh            ;finished
                579
0000022f        580 finscc:
0000022f e1     581             pop     hl              ;
00000230 c1     582             pop     bc              ;
00000231 f1     583             pop     af              ;
00000232 c9     584             ret
                585
```

**LISTING 2**

```
                        600                                     ;************************************************
                        601                                     ;Subroutine to transmit the llapenq packet
                        602                                     ;************************************************
00000244                603 txenq:
                        604
00000244 f5             605             push    af              ;save status and a reg
00000245 c5             606             push    bc              ;save
00000246 e5             607             push    hl              ;save
                        608                                     ;
00000247 f3             609             di                      ;make sure that
                        610                                     ;no interrupt routine
                        611                                     ;nor should interrupt
                        612                                     ;occur during
                        613                                     ;this subroutine.
00000248 3e03           614             ld      a,03h
0000024a d3e8           615             out     (scc_cont),a    ;WR3
0000024c 3ecc           616             ld      a,0cch
0000024e d3e8           617             out     (scc_cont),a    ;8b/char,rx crc
                        618                                     ;enable,addrs src
                        619                                     ;and rx disabled
                        620
00000250 3e0a           621             ld      a,0ah           ;select WR10
00000252 d3e8           622             out     (scc_cont),a
00000254 3ee0           623             ld      a,11100000b     ;idle with flags
00000256 d3e8           624             out     (scc_cont),a
                        625
                        626                                     ;****enable transmitter *****
00000258 3e05           627             ld      a,05h           ;select WR5
0000025a d3e8           628             out     (scc_cont),a
0000025c 3e68           629             ld      a,01101000b     ;enable tx
0000025e d3e8           630             out     (scc_cont),a
                        631                                     ;
                        632                                     ;
                        633                                     ;****enable rs-422 driver *****
00000260 3e05           634             ld      a,05h           ;select WR5
00000262 d3e8           635             out     (scc_cont),a
00000264 3e6a           636             ld      a,01101010b     ;enable tx,
00000266 d3e8           637             out     (scc_cont),a    ;reset rts
00000268 00             638             nop
00000269 00             639             nop
                        640                                     ;nop's needed to complete 4.3 usec
                        641                                     ;for 1 bit time enable of transmitter.
                        642                                     ;total delay=2*(7+11+4) T states at 10 MHZ
                        643                                     ;
                        644                                     ;****disable rs-422 driver for 2 bit times*****
0000026a 3e05           645             ld      a,05h           ;select WR5
0000026c d3e8           646             out     (scc_cont),a
0000026e 3e68           647             ld      a,01101000b     ;enable tx, set rts
00000270 d3e8           648             out     (scc_cont),a
                        649                                     ;
00000272 3e80           650             ld      a,10000000b     ;reset txcrc
00000274 d3e8           651             out     (scc_cont),a
00000276 0601           652             ld      b,01h           ;delay count
00000278                653 csloop:
00000278 10fe           654             djnz    csloop          ;loop needed
                        655                                     ;to complete
                        656                                     ;8.6 usec  min.
                        657                                     ;or 2 bit times.
                        658                                     ;****enable rs-422 driver for llap transmission*****
0000027a 3e05           659             ld      a,05h           ;select WR5
```

```
0000027c d3e8          660             out     (scc_cont),a
0000027e 3e6b          661             ld      a,01101011b     ;sdlc crc,
                       662                                     ;txcrc enable,
                       663                                     ;reset rts
00000280 d3e8          664             out     (scc_cont),a
                       665
                       666                                      ;
                       667                                     ;**start counting out 2 flag character times **
                       668                                      ;
                       669                                     ;count 16 bit times
                       670                                     ;from the rs-422 enable
                       671                                     ;for 2 flags.
                       672                                     ;btdelay=subr delay+ctc1int+polling=8bits
                       673                                     ;16 bit times-btdelay=16-8=08h
                       674                                     ;
00000282 0e08          675             ld      c,08h
00000284 cdWwww        676             call    bittime         ;bittime delay
                       677                                     ;is stored in reg.c
                       678                                     ;and bit1 of timflg
                       679                                     ;will indicate
                       680                                     ;count termination.
                       681
00000287682 l6:                                                ;timer flag
00000287 3aWwww        683             ld      a,(timflg)      ;
0000028a cb4f          684             bit     1,a             ;if bit1=1 then
                       685                                     ;count terminated
0000028c 28f9          686             jr      z,l6            ;
0000028e cb8f          687             res     1,a             ;reset timflg bit1
00000290 32Wwww        688             ld      (timflg),a      ;update timflg
                       689                                      ;
00000293 3e03          690             ld      a,03h           ;
00000295 d3e5          691             out     (ctc1_cont),a   ;disable int,
                       692                                     ;software reset
                       693                                     ;to kill the counter1
00000297 0602          694             ld      b,02h           ;2+1 bytes to transmitted
00000299 21Wwww        695             ld      hl,txlapenq     ;point to txlapenq buffer
                       696                                     ;send 1st byte
0000029c 7e            697             ld      a,(hl)          ;
0000029d d3e9          698             out     (scc_data),a    ;and send it
0000029f 23            699             inc     hl              ;point to the next byte
                       700                                      ;
000002a0 3ec0          701             ld      a,0c0h          ;reset eom latch command
000002a2 d3e8          702             out     (scc_cont),a
                       703                                     ;
000002a4 f3            704             di                      ;disable all int
000002a5 3e0a          705             ld      a,0ah           ;select WR10
000002a7 d3e8          706             out     (scc_cont),a
                       707                                     ;idle with 1's
                       708                                     ;at the end of the frame
000002a9 3ee8          709             ld      a,11101000b
000002ab d3e8          710             out     (scc_cont),a
                       711                                     ;
000002ad 3e00          712 txq2:       ld      a,00h
000002af d3e8          713             out     (scc_cont),a    ;rr0
000002b1 dbe8          714             in      a,(scc_cont)    ;read rr0
000002b3 cb57          715             bit     2,a             ;read tx buffer empty
000002b5 28f6          716             jr      z,txq2          ;loop if zero
000002b7717 txq1:
000002b7 7e            718             ld      a,(hl)          ;
000002b8 d3e9          719             out     (scc_data),a    ;and send it
000002ba 23            720             inc     hl              ;point to the next byte
                       721
```

```
000002bb 10f0    722        djnz    txq2          ;loop until all
                 723                               ;bytes have been
                 724                               ;transmitted.
                 725
                 726
000002bd 3e28    727        ld      a,028h        ;reset tx int pending
000002bf d3e8    728        out     (scc_cont),a
                 729                               ;note:tx buffer
                 730                               ;empty happens as tx
                 731                               ;shifter is loaded.
                 732                                ;
                 733                               ;count= last byte+
                 734                               ;crc+flag+12bit times-btdelay
                 735                               ;btdelay=subr delay+ctc1int+polling=8bits
                 736                               ;8+16+8+12-8=36=24h
000002c1 0e24    737        ld      c,24h
000002c3 cdWwww  738        call    bittime       ;bittime delay
                 739                               ;is stored in reg.c
                 740                                ;
000002c6741 l7:                                    ;timer flag
000002c6 3aWwww  742        ld      a,(timflg)    ;
000002c9 cb4f    743        bit     1,a           ;if bit1=1 then count finish
000002cb 28f9    744        jr      z,l7          ;
000002cd cb8f    745        res     1,a           ;reset timflg bit1
000002cf 32Wwww  746        ld      (timflg),a    ;update timflg
                 747                               ;
000002d2 3e03    748        ld      a,03h         ;
000002d4 d3e5    749        out     (ctc1_cont),a ;disable int,software reset
                 750                               ;to kill counter
                 751                               ;****disable rs-422 driver after 12 to 18 1's*****
000002d6 3e05    752        ld      a,05h         ;select WR5
000002d8 d3e8    753        out     (scc_cont),a
000002da 3e60    754        ld      a,01100000b   ;disable tx, set rts
000002dc d3e8    755        out     (scc_cont),a
                 756
000002de 3e03    757        ld      a,03h
000002e0 d3e8    758        out     (scc_cont),a  ;WR3
000002e2 3ecd    759        ld      a,0cdh
000002e4 d3e8    760        out     (scc_cont),a  ;8b/char,rx crc enabled,
                 761                               ;address search and rx enabled
                 762
                 763                               ;************************************
                 764
                 765                               ;count for the interframe gap
                 766                               ;of 200 usec or 46 bit times.
                 767                               ;btdelay=subr delay+ctc1int+polling=8bits
                 768                               ;46 - btdelay=46-8=26h
                 769                               ;note that timflg will be polled in
                 770                               ;the main routine.
                 771                               ;
000002e6 0e26    772        ld      c,26h
000002e8 cdWwww  773        call    bittime
                 774                               ;
                 775                               ;bittime delay is stored in reg.c
                 776                               ;************************************
000002eb e1      777        pop     hl            ;restore
000002ec c1      778        pop bc                ;restore
000002ed f1      779        pop     af            ;restore status and a reg
000002ee c9      780        ret
                 781
                 782
                 783
```

```
                        784                                     ;subroutine to time out bit time 4.3 usec per bit
                        785                                     ;register c contains the number of bits to be
                                                                counted down
                        786                                     ;***********************************************
000002ef                787 bittime:
000002ef f5             788             push    af              ;save status and a reg
000002f0 c5             789             push    bc              ;save
000002f1 e5             790             push    hl              ;save
                        791                                     ;
000002f2 3ed2           792             ld      a,0d2h          ;ctc1 int vector
000002f4 d3e5           793             out     (ctc1_cont),a
                        794
000002f6 3edf           795             ld      a,11011111b     ;
000002f8 d3e5           796             out     (ctc1_cont),a   ;enable int
                        797                                     ;select counter mode
                        798                                     ;clk/trg edge starts with rising edg
                        799                                     ;time constant follows
                        800                                     ;software reset
000002fa 79             801             ld      a,c             ;reg c contains the number of bits
000002fb d3e5           802             out     (ctc1_cont),a   ;load the number of bits to be counted
                        803                                     ;**
000002fd e1             804             pop     hl              ;restore
000002fe c1             805             pop     bc              ;restore
000002ff f1             806             pop     af              ;restate status and a reg
00000300 fb             807             ei
00000301 c9             808             ret
                        809
```

**LISTING 3**

```
                        1131                                    ;******************************
                        1132                                    ;receive int service routine.
                        1133                                    ;******************************
                        1134                                    ;save received character in receiver buffer
                        1135                                    ;to by rxpointer
                        1136
0000044d                1137recint:
0000044d f5             1138            push    af              ;save af
0000044e d5             1139            push    de
0000044f e5             1140            push    hl
00000450 dbe9           1141            in      a,(scc_data)    ;read scc data
00000452 2aWwww         1142            ld      hl,(rxpointer)  ;
00000455 77             1143            ld      (hl),a          ;save it
00000456 23             1144            inc     hl              ;update pointer
00000457 22Www          1145            ld      (rxpointer),hl  ;
0000045a ed5bWwww       1146            ld      de,(rxbufend)   ;end of rx buffer
0000045e af             1147            xor     a               ;reset cy
0000045f ed52           1148            sbc     hl,de           ;
00000461 c2Wwww         1149            jp      nz,recexit      ;if not zero,then receive
                                                                byte length is ok
00000464 21Wwww         1150            ld      hl,recerrflg    ;
00000467 cbc6           1151            set     0,(hl)          ;set bit0=1 maxfrmflg to indicate error
                        1152                                    ;because of max frame
                                                                size exceeded.
00000469                1153recexit:
00000469 3e38           1154            ld      a,038h
0000046b d3e8           1155            out     (scc_cont),a    ;reset highest ius
0000046d e1             1156            pop     hl
0000046e d1             1157            pop     de
0000046f f1             1158            pop     af              ;restore af
00000470 fb             1159            ei                      ;enable int
00000471 c9             1160            ret                     ;return from int
                        1161                                    ;note ret and not reti is used for scc
                        1162                                    ;interrupts on the z80181.
                        1163
                        1164                                    ;*********************************************
                        1165                                    ;special receive interrupt service routine
                        1166                                    ;*********************************************
                        1167;                                   “parity is special condition” bit is off.
                        1168;                                   special conditions are eof or rx overrun error.
                        1169;                                   crc error flag is valid only if eof is valid.
                        1170;                                   if frame is ok then recerrflg bit1=0, otherwise
                        1171
00000472                1172 spcond:
00000472 f5             1173            push    af              ;save af reg
00000473 c5             1174            push    bc              ;
00000474 e5             1175            push    hl;
                        1176
00000475 3e01           1177            ld      a,01h
00000477 d3e8           1178            out     (scc_cont),a    ;read rr1
00000479 dbe8           1179            in      a,(scc_cont)
0000047b e660           1180            and     01100000b        ;check bit6 (crc) or bit5 (overrun)
0000047d caWwww         1181            jp      z,ok            ;
                        1182            ;
00000480 21Wwww         1183            ld      hl,recerrflg    ;fetch receive error flag
00000483 cbce           1184            set     1,(hl)          ;set bit1=1 for frame not ok
00000485 c3Wwww         1185            jp      crc_exit
```

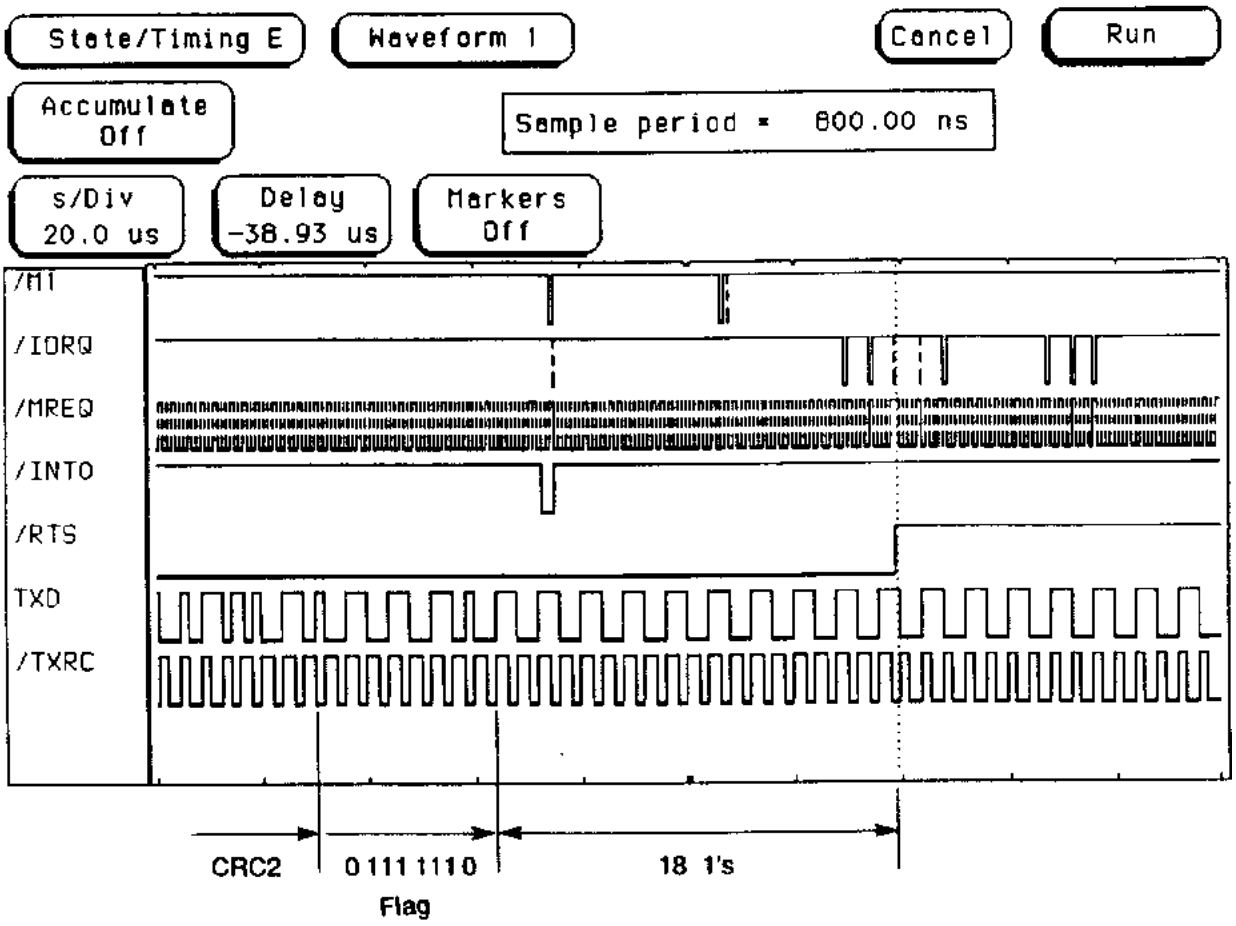
```
00000488                1186            ok:
00000488 21Wwww         1187            ld      hl,recerrflg    ;fetch receive error flag
0000048b cb8            1188            res     1,(hl)          ;set bit1=0 for frame  ok
                        1189
                        1190
0000048d                1191 crc_exit:
0000048d dbe9           1192            in      a,(scc_data)    ;read 2nd crc (debug only) and
0000048f 2aWwww         1193            ld      hl,(rxpointer)  ;load pointer
00000492 2b             1194            dec     hl              ;adjust rx buff ptr for crc1
00000493 2b             1195            dec     hl              ;adjust rx buff ptr for crc2
00000494 22Wwww         1196            ld      (rxpointer),hl  ;
                        1197
00000497                1198 spexit:
00000497 3e38           1199            ld      a,038h
00000499 d3e8           1200            out     (scc_cont),a    ;reset highest ius
                        1201
0000049b e1             1202            pop     hl              ;restore hl
0000049c c1             1203            pop     bc              ;restore be
0000049d f1             1204            pop     af              ;restore af
0000049e fb             1205            ei                      ;enable int
0000049f c9             1206            ret                     ;return from int
                        1207
```

**LISTING 4**

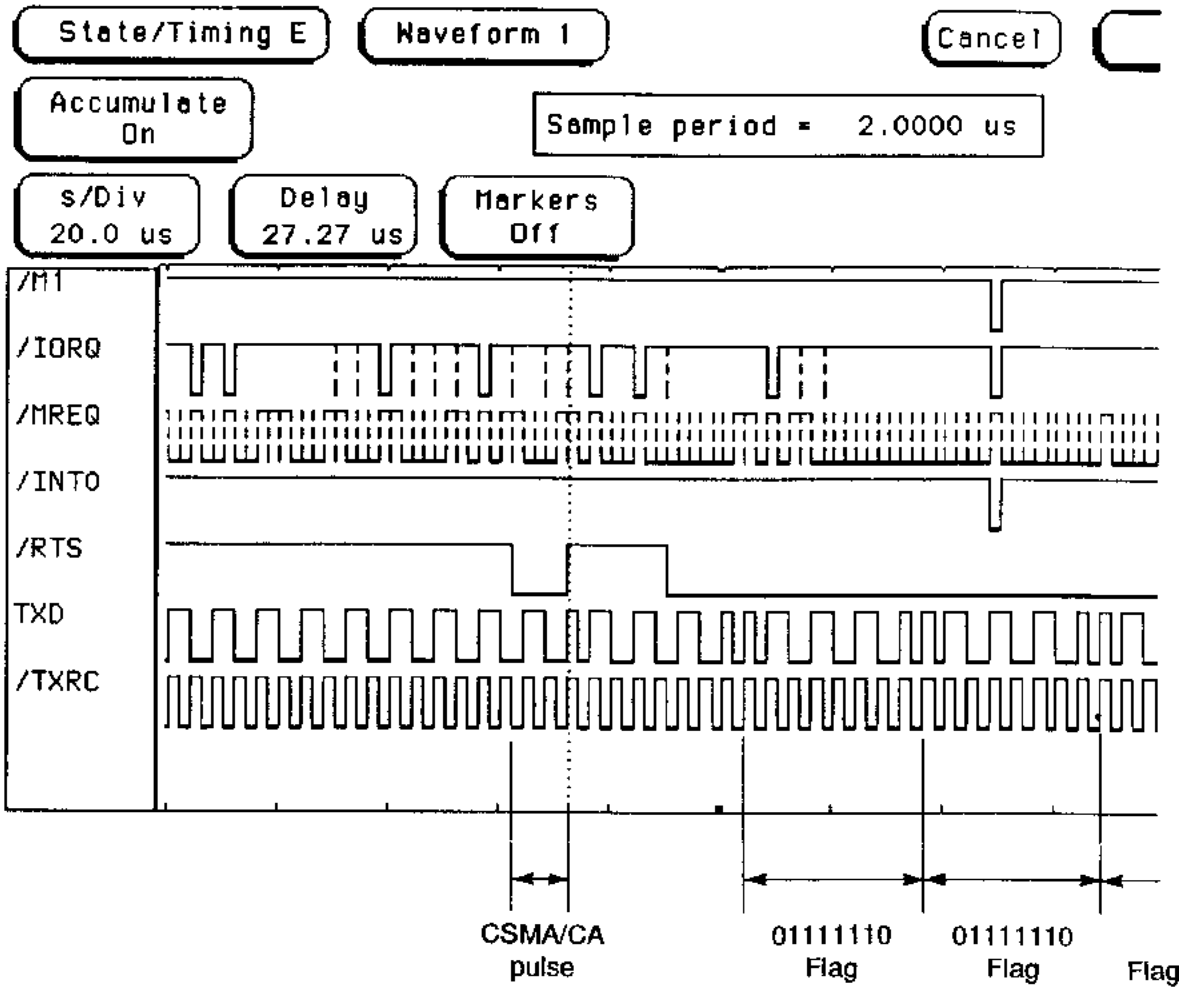
```
                        1306
                        1307                    ;****************************************
                        1308                    ;ctc1 timer int handler
                        1309                    ;****************************************
00000509                1310 ctc1int:
                        1311                    ;ctc1 is programmed in counter mode.
                        1312                    ;external trigger edges is provided by
                        1313                    ;/trxc pin at intervals of 4.3 usec.
                        1314                    ;bit1 of timflg is set when count is terminated.
00000509 f5             1315            push    af
0000050a c5             1316            push    bc
0000050b e5             1317            push    hl
                        1318                    ;** update the timing flag **
0000050c 21Wwww         1319            ld      hl,timflg
0000050f 7e             1320            ld      a,(hl)          ;get recent timflg
00000510 cbcf           1321            set     1,a             ;bit1=1 after count is over
00000512 77             1322            ld      (hl),a          ;update the timflg
00000513 e1             1323            pop     hl
00000514 c1             1324            pop     bc
00000515 f1             1325            pop     af
00000516 fb             1326            ei
00000517 ed4d           1327            reti
                        1328
                        1329
                        1330
                        1331                    ;**********************************
                        1332                    ;interrupt vector table for the scc
                        1333                    ;**********************************
                        1334                    ;the status of the interrupt source will affect
                        1335                    ;the interrupt vector. The interrupt handler's
                        1336                    ;address are set in a block, as below.
00000a00                1337            org     sdlc + 0a00h
00000a00                1338 sccvect:
                        1339            if      scc_a
00000a00                1340            .block  8               ;reserve vector for other ch
                        1341            endif
00000a08 R000+03e9,     1342            dw      txint           ;tx int
00000a0a R000+04c8,     1343            dw      ext_stat        ;ext/stat int
00000a0c R000+0433,     1344            dw      recint          ;rx char int
00000a0e R000+0454,     1345            dw      spcond          ;sp rec cond int
                        1346
                        1347            if      not scc_a
00000a10                1348            .block  8               ;reserve vector for other ch
                        1349            endif
                        1350
00000a18                1351 temp:      .block  1
                        1352
                        1353
                        1354                    ;**********************************
                        1355                    ;interrupt vector table for the ctc
                        1356                    ;**********************************
00000ad0                1357            org     0ad0h
00000ad0 R000+04d8,     1358            dw      ctc0int         ;reserved for ctc0 int routine
00000ad2                1359            org     0ad2h
00000ad2 R000+0509,     1360            dw      ctc1int         ;reserved for ctc1 int routine
                        1361
                        1362                    ;************************
                        1363                    ;receive buffer area
                        1364                    ;************************
```

```
00001000                1365            org             1000h
00001000                1366 rx_buff:   .block          length
                        1367
                        1388
                        1389                                    ;************************
                        1390                                    ;transmitter buffer area
                        1391                                    ;************************
0000b000                1392            org             0b000h
                        1398                                    ;
                        1399                                    ;*********************************
                        1400                                    ;transmit llap enq packet (3bytes)
                        1401                                    ;*********************************
0000b258 ff             1402 txlapenq:  db              0ffh    ;broadcast id
0000b259                1403            .block          1       ;guess at myaddress
                        myaddress
0000b25a 81             1404            db              81h     ;llap enq type
                        1405                                    ;
```

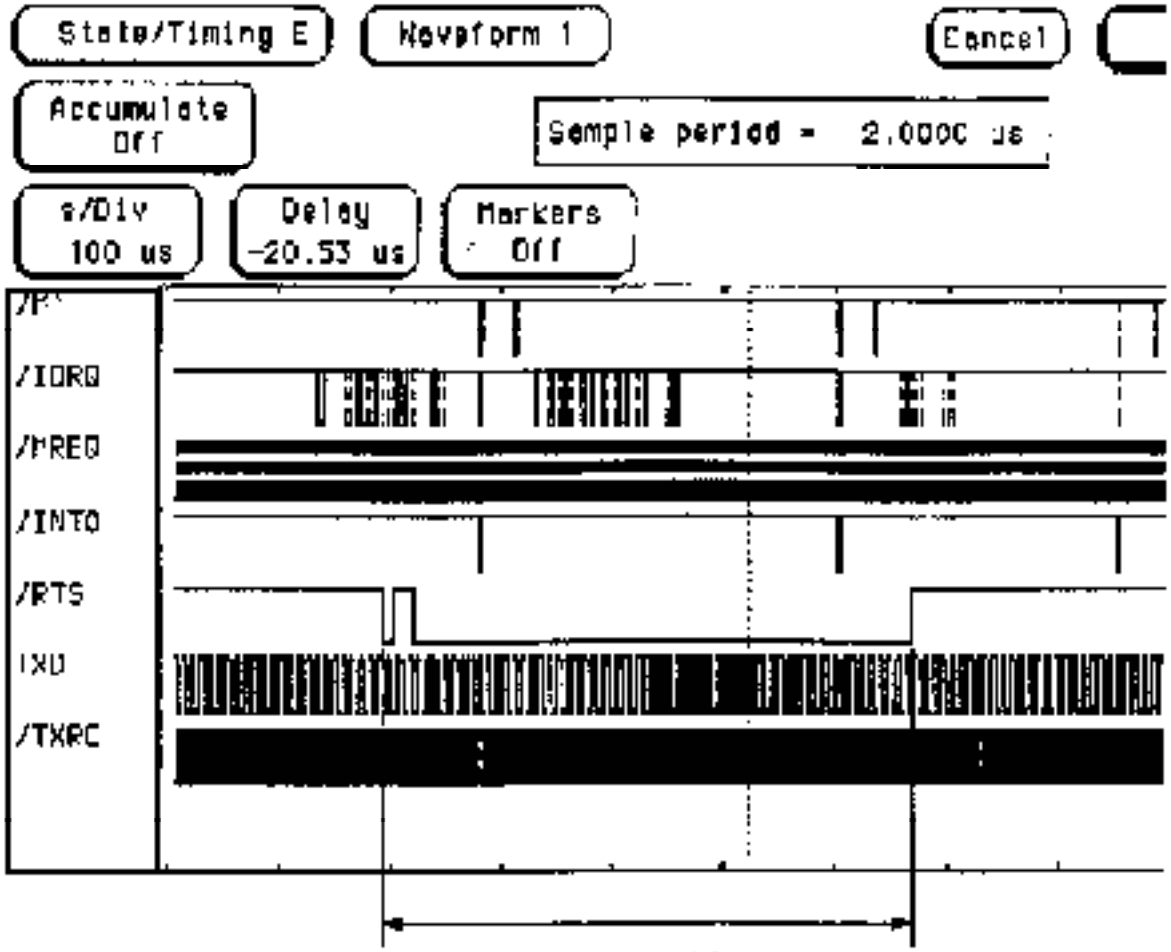
## APPENDIX B




**12 to 18 1's at the end of an LLAP frame**




**CSMA/CA before an LLAP frame**




**An LLAP Frame**

# ON-CHIP OSCILLATOR DESIGN

*Design and build reliable, cost-effective, on-chip oscillator circuits that are trouble free. PUTTING OSCILLATOR THEORY INTO A PRACTICAL DESIGN MAKES FOR A MORE DEPENDABLE CHIP.*

## INTRODUCTION

This Application Note (App Note) is written for designers using Zilog Integrated Circuits with on-chip oscillators; circuits in which the amplifier portion of a feedback oscillator is contained on the IC. This App Note covers common theory of oscillators, and requirements of the circuitry (both internal and external to the IC) which comes from the theory for crystal and ceramic resonator based circuits.

### Purpose and Benefits

The purposes and benefits of this App Note include:

1. Providing designers with greater understanding of how oscillators work and how to design them to avoid problems.
2. To eliminate field failures and other complications resulting from an unawareness of critical on-chip oscillator design constraints and requirements.

### Problem Background

Inadequate understanding of the theory and practice of oscillator circuit design, especially concerning oscillator startup, has resulted in an unreliable design and subsequent field problems (See on page 10 for reference materials and acknowledgments).

## OSCILLATOR THEORY OF OPERATION

The circuit under discussion is called the Pierce Oscillator (Figures 1, 2). The configuration used is in all Zilog on-chip oscillators. Advantages of this circuit are low power consumption, low cost, large output signal, low power level in the crystal, stability with respect to $V_{CC}$ and temperature, and low impedances (not disturbed by stray effects). One drawback is the need for high gain in the amplifier to compensate for feedback path losses.

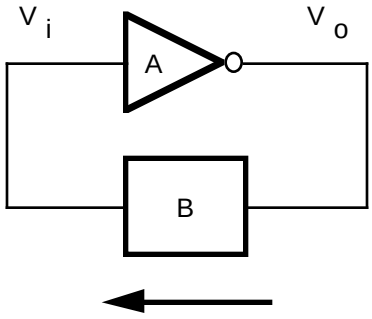



**Figure 1. Basic Circuit and Loop Gain**

## OSCILLATOR THEORY OF OPERATION (Continued)

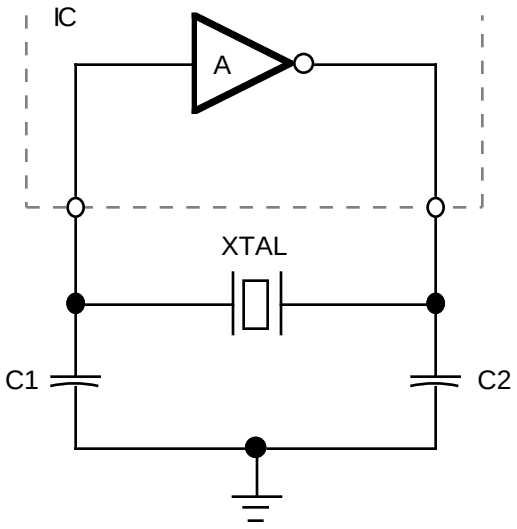



**Figure 2. Zilog Pierce Oscillator**

### Pierce Oscillator (Feedback Type)

The basic circuit and loop gain is shown in Figure 1. The concept is straightforward; gain of the amplifier is A = Vo/Vi. The gain of the passive feedback element is B = Vi/Vo. Combining these equations gives the equality AB = 1. Therefore, the total gain around the loop is unity. Also, since the gain factors A and B are complex numbers, they have phase characteristics. It is clear that the total phase shift around the loop is forced to zero (i.e., 360 degrees), since $V_{IN}$ must be in phase with itself. In this circuit, the amplifier ideally provides 180 degrees of phase shift (since it is an inverter). Hence, the feedback element is forced to provide the other 180 degrees of phase shift.

Additionally, these gain and phase characteristics of both the amplifier and the feedback element vary with frequency. Thus, the above relationships must apply at the frequency of interest. Also, in this circuit the amplifier is an active element and the feedback element is passive. Thus, by definition, the gain of the amplifier at frequency must be greater than unity, if the loop gain is to be unity.

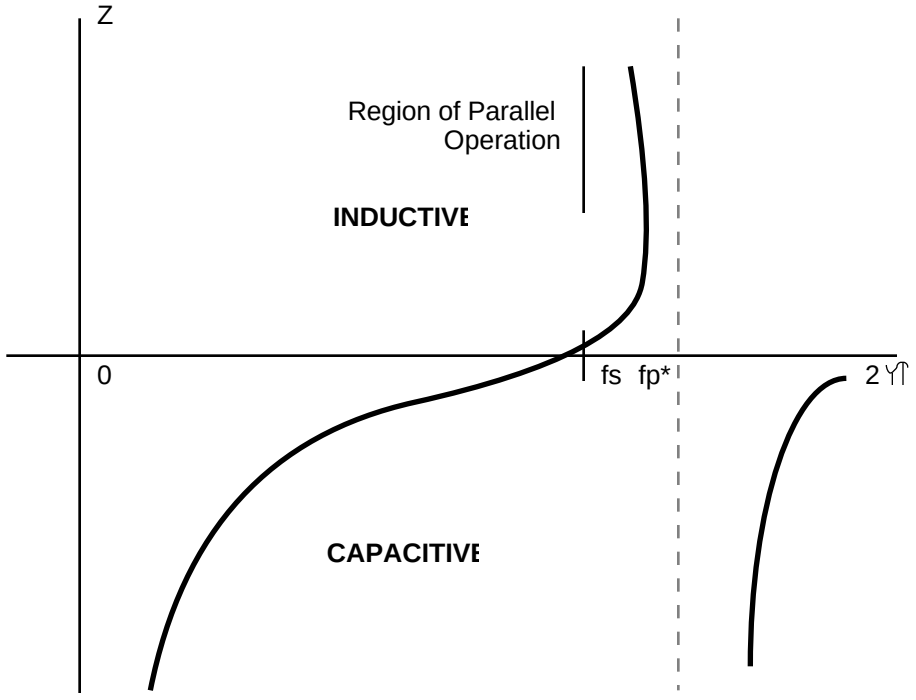
The described oscillator amplifies its own noise at startup until it settles at the frequency which satisfies the gain/phase requirement AB = 1. This means loop gain equals one, and loop phase equals zero (360 degrees). To do this, the loop gain at points around the frequency of oscillation must be greater than one. This achieves an average loop gain of one at the operating frequency.

The amplifier portion of the oscillator provides gain > 1 plus 180 degrees of phase shift. The feedback element provides the additional 180 degrees of phase shift without attenuating the loop gain to < 1. To do this the feedback element is inductive, i.e., it must have a positive reactance at the frequency of operation. The feedback elements discussed are quartz crystals and ceramic resonators.

### Quartz Crystals

A quartz crystal is a piezoelectric device; one which transforms electrical energy to mechanical energy and vice versa. The transformation occurs at the resonant frequency of the crystal. This happens when the applied AC electric field is sympathetic in frequency with the mechanical resonance of the slice of quartz. Since this characteristic can be made very accurate, quartz crystals are normally used where frequency stability is critical. Typical frequency tolerance is .005 to 0.3%.

The advantage of a quartz crystal in this application is its wide range of positive reactance values (i.e., it looks inductive) over a narrow range of frequencies (Figure 3).




* fs - fp is very small (approximately 300 parts per million)

**Figure 3. Series vs. Parallel Resonance**

However, there are several ranges of frequencies where the reactance is positive; these are the fundamental (desired frequency of operation), and the third and fifth mechanical overtones (approximately 3 and 5 times the fundamental frequency). Since the desired frequency range in this application is always the fundamental, the overtones must be suppressed. This is done by reducing the loop gain at these frequencies. Usually, the amplifier's gain roll off, in combination with the crystal parasitics and load capacitors, is sufficient to reduce gain and prevent oscillation at the overtone frequencies.

The following parameters are for an equivalent circuit of a quartz crystal (Figure 4):

**L** - motional inductance (typ 120 mH @ 4 MHz)

**C** - motional capacitance (typ .01 pf @ 4 MHz)

**R** - motional resistance (typ 36 ohm @ 4 MHz)

**Cs** - shunt capacitance resulting from the sum of the capacitor formed by the electrodes (with the quartz as a dielectric) and the parasitics of the contact wires and holder (typ 3 pf @ 4 MHz).

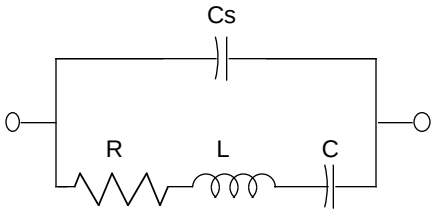
The series resonant frequency is given by:

**Fs** = 1/(2π x sqrt of LC),
where Xc and Xl are equal.

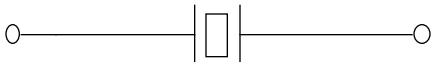
Thus, they cancel each other and the crystal is then R shunted by Cs with zero phase shift.

The parallel resonant frequency is given by:

**Fp** = 1/[2π x sqrt of L (C Ct/C+Ct)],
where: $Ct = C_L + C_S$




Quartz Equivalent Circuit



Symbolic Representation

**Figure 4. Quartz Oscillator**

## OSCILLATOR THEORY OF OPERATION (Continued)

**Series vs. Parallel Resonance.** There is very little difference between series and parallel resonance frequencies (Figure 3). A series resonant crystal (operating at zero phase shift) is desired for non-inverting amplifiers. A parallel resonant crystal (operating at or near 180 degrees of phase shift) is desired for inverting amps. Figure 3 shows that the difference between these two operating modes is small. Actually, all crystals have operating points in both serial and parallel modes. A series resonant circuit will NOT have load caps C1 and C2. A data sheet for a crystal designed for series operation does not have a load cap spec. A parallel resonant crystal data sheet specifies a load cap value which is the series combination of C1 and C2. For this App Note discussion, since all the circuits of interest are inverting amplifier based, only the parallel mode of operation is considered.

### Ceramic Resonators

Ceramic resonators are similar to quartz crystals, but are used where frequency stability is less critical and low cost is desired. They operate on the same basic principle as quartz crystals as they are piezoelectric devices and have a similar equivalent circuit. The frequency tolerance is wider (0.3 to 3%), but the ceramic costs less than quartz. Figure 5 shows reactance vs. frequency and Figure 6 shows the equivalent circuit.

Typical values of parameters are L = .092 mH, C = 4.6 pf, R = 7 ohms and Cs = 40 pf, all at 8 MHz. Generally, ceramic resonators tend to start up faster but have looser frequency tolerance than quartz. This means that external circuit parameters are more critical with resonators.

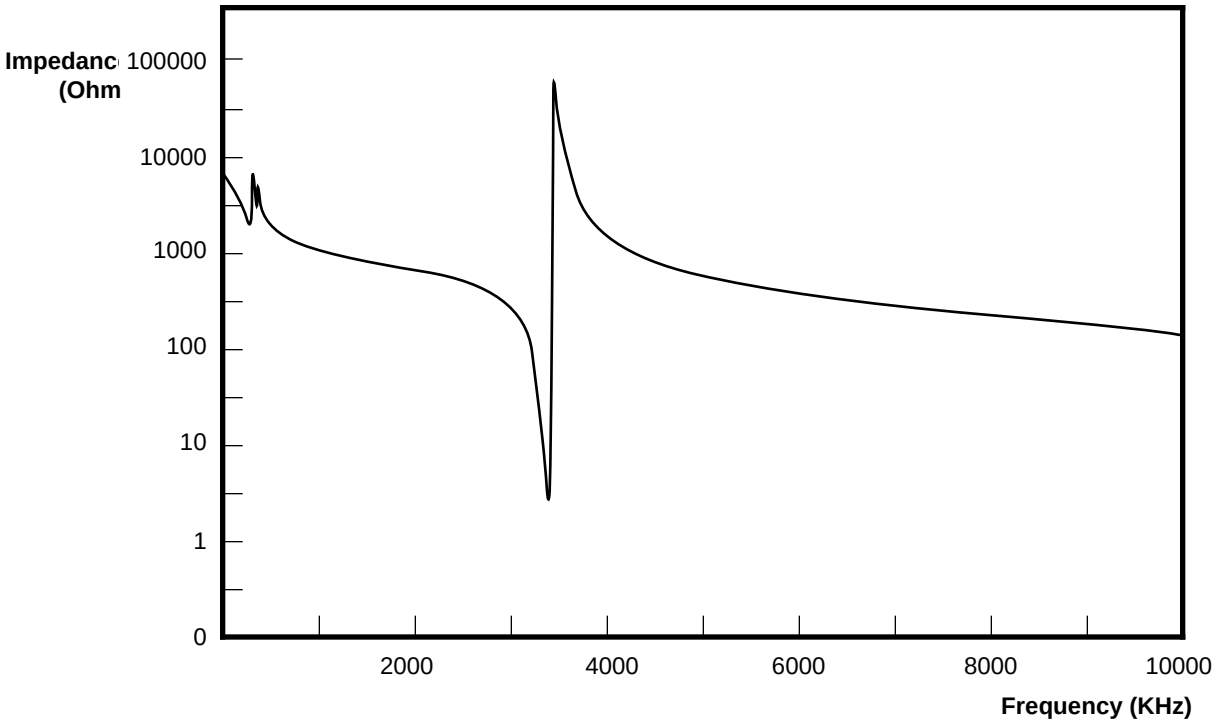



**Figure 5. Ceramic Resonator Reactance**

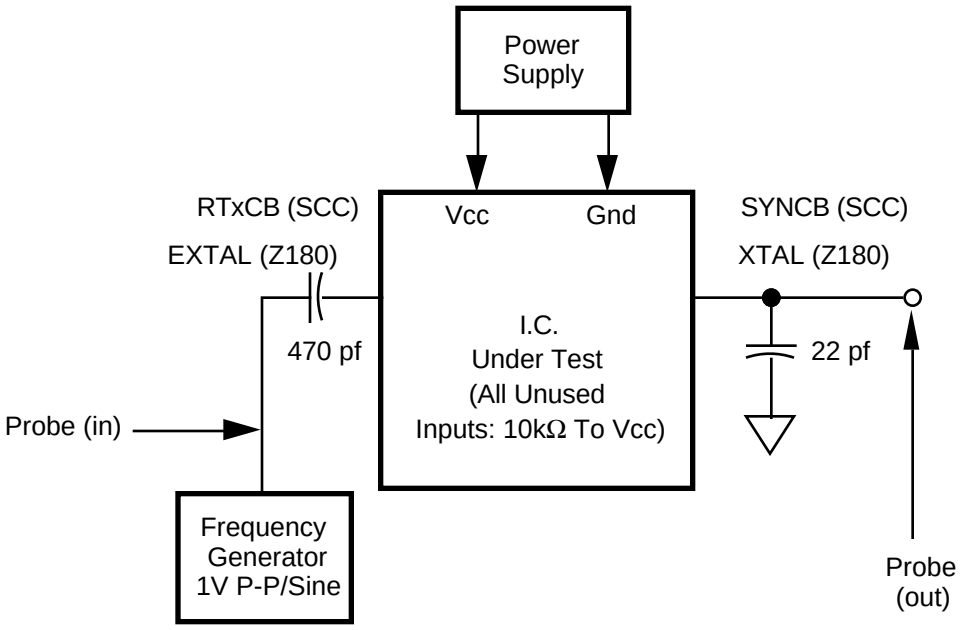



**Figure 6. Gain Measurement**

## Load Capacitors

The effects/purposes of the load caps are:

Cap C2 combined with the amp output resistance provides a small phase shift. It also provides some attenuation of overtones.

Cap C1 combined with the crystal resistance provides additional phase shift.

These two phase shifts place the crystal in the parallel resonant region of Figure 3.

Crystal manufacturers specify a load capacitance number. This number is the load seen by the crystal which is the series combination of C1 and C2, including all parasitics (PCB and holder). This load is specified for crystals meant to be used in a parallel resonant configuration. The effect on startup time; if C1 and C2 increase, startup time increases to the point at which the oscillator will not start. Hence, for fast and reliable startup, over manufacture of large quantities, the load caps should be sized as low as possible without resulting in overtone operation.

## Amplifier Characteristics

The following text discusses open loop gain vs. frequency, open loop phase vs. frequency, and internal bias.

**Open Loop Gain vs. Frequency over lot, VCC, Process Split, and Temp.** Closed loop gain must be adequate to start the oscillator and keep it running at the desired frequency. This means that the amplifier open loop gain must be equal to one plus the gain required to overcome the losses in the feedback path, across the frequency band and up to the frequency of operation. This is over full process, lot, $V_{CC}$, and temperature ranges. Therefore, measuring the open loop gain is not sufficient; the losses in the feedback path (crystal and load caps) must be factored in.

**Open Loop Phase vs. Frequency.** Amplifier phase shift at and near the frequency of interest must be 180 degrees plus some, minus zero. The parallel configuration allows for some phase delay in the amplifier. The crystal adjusts to this by moving slightly down the reactance curve (Figure 3).

**Internal Bias.** Internal to the IC, there is a resistor placed from output to input of the amplifier. The purpose of this feedback is to bias the amplifier in its linear region and to provide the startup transition. Typical values are 1M to 20M ohms.

## PRACTICE: CIRCUIT ELEMENT AND LAY OUT CONSIDERATIONS

The discussion now applies prior theory to the practical application.

### Amplifier and Feedback Resistor

The elements of the circuit, internal to the IC, include the amplifier, feedback resistor, and output resistance. The amplifier is modeled as a transconductance amplifier with a gain specified as $I_{OUT}/V_{IN}$ (amps per volt).

**Transconductance/Gain.** The loop gain AB = gm x Z1, where gm is amplifier transconductance (gain) in amps/volt and Z1 is the load seen by the output. AB must be greater than unity at and about the frequency of operation to sustain oscillation.

**Gain Measurement Circuit.** The gain of the amplifier can be measured using the circuits of Figures 6 & 7. This may be necessary to verify adequate gain at the frequency of interest and in determining design margin.

**Gain Requirement vs. Temperature, Frequency and Supply Voltage.** The gain to start and sustain oscillation (Figure 8) must comply with:

$gm > 4\pi^2 f^2 Rq C_{IN} C_{OUT}t \times M$
where:
M is a quartz form factor = $(1 + C_{OUT}/C_{IN} + C_{OUT}/C_{OUT})_2$

**Output Impedance.** The output impedance limits power to the XTAL and provides small phase shift with load cap C2.

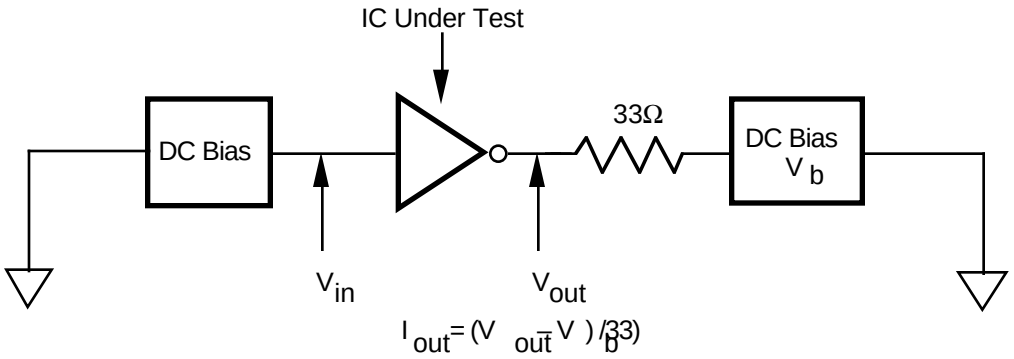
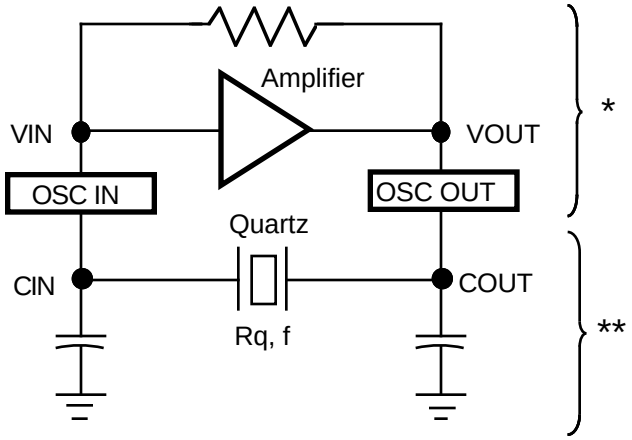



**Figure 7. Transconductance (gm) Measurement**




* Inside chip, feedback resistor biases the amplifier in the high gm region.

** External components typically: CIN = COUT = 30 to 50 pf (add 10 pf pin cap).

**Figure 8. Quartz Oscillator Configuration**

## Load Capacitors

In the selection of load caps it is understood that parasitics are always included.

**Upper Limits.** If the load caps are too large, the oscillator will not start because the loop gain is too low at the operating frequency. This is due to the impedance of the load capacitors. Larger load caps produce a longer startup.

**Lower Limits.** If the load caps are too small, either the oscillator will not start (due to inadequate phase shift around the loop), or it will run at a 3rd, 5th, or 7th overtone frequency (due to inadequate suppression of higher overtones).

**Capacitor Type and Tolerance.** Ceramic caps of ±10% tolerance should be adequate for most applications.

**Ceramic vs. Quartz.** Manufacturers of ceramic resonators generally specify larger load cap values than quartz crystals. Quartz C is typically 15 to 30 pF and ceramic typically 100 pF.

**Summary.** For reliable and fast startup, capacitors should be as small as possible without resulting in overtone operation. The selection of these capacitors is critical and all of the factors covered in this note should be considered.

## Feedback Element

The following text describes the specific parameters of a typical crystal:

**Drive Level.** There is no problem at frequencies greater than 1 MHz and $V_{CC}$ = 5V since high frequency AT cut crystals are designed for relatively high drive levels (5-10 mw max).

A typical calculation for the approximate power dissipated in a crystal is:

$$P = 2R (\pi \times f \times C \times V_{CC})_2$$

**Where.** R = crystal resistance of 40 ohms, C = C1 + Co = 20 pF. The calculation gives a power dissipation of 2 mW at 16 MHz.

**Series Resistance.** Lower series resistance gives better performance but costs more. Higher R results in more power dissipation and longer startup, but can be compensated by reduced C1 and C2. This value ranges from 200 ohms at 1 MHz down to 15 ohms at 20 MHz.

**Frequency.** The frequency of oscillation in parallel resonant circuits is mostly determined by the crystal (99.5%). The external components have a negligible effect (0.5%) on frequency. The external components (C1,C2) and layout are chosen primarily for good startup and reliability reasons.

**Frequency Tolerance (initial temperature and aging).** Initial tolerance is typically ±.01%. Temperature tolerance is typically ±.005% over the temp range (-30 to +100 degrees C). Aging tolerance is also given, typically ±.005%.

**Holder.** Typical holder part numbers are HC6, 18, 25, 33, 44.

**Shunt Capacitance.** (Cs) typically <7 pf.

**Mode.** Typically the mode (fundamental, 3rd or 5th overtone) is specified as well as the loading configuration (series vs. parallel).

The ceramic resonator equivalent circuit is the same as shown in Figure 4. The values differ from those specified in the theory section. Note that the ratio of L/C is much lower than with quartz crystals. This gives a lower Q which allows a faster startup and looser frequency tolerance (typically ±0.9% over time and temperature) than quartz.

## Layout

The following text explains trace layout as it affects the various stray capacitance parameters (Figure 9).

**Traces and Placement.** Traces connecting crystal, caps, and the IC oscillator pins should be as short and wide as possible (this helps reduce parasitic inductance and resistance). Therefore, the components (caps and crystal) should be placed as close to the oscillator pins of the IC as possible.

**Grounding/Guarding.** The traces from the oscillator pins of the IC should be guarded from all other traces (clock, $V_{CC}$, address/data lines) to reduce crosstalk. This is usually accomplished by keeping other traces away from the oscillator circuit and by placing a ground ring around the traces/components (Figure 9).

## Measurement and Observation

Connection of a scope to either of the circuit nodes is likely to affect operation because the scope adds 3-30 pF of capacitance and 1M-10M ohms of resistance to the circuit.

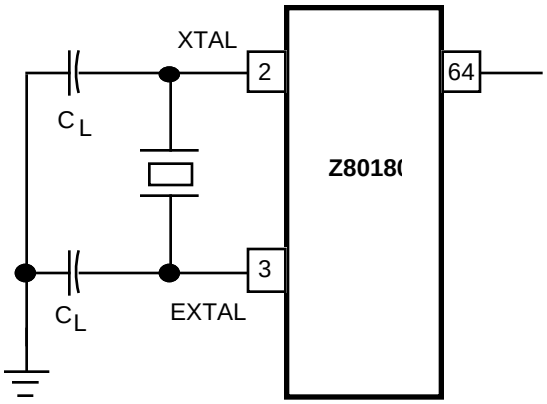
## Indications of an Unreliable Design

There are two major indicators which are used in working designs to determine their reliability over full lot and temperature variations. They are:

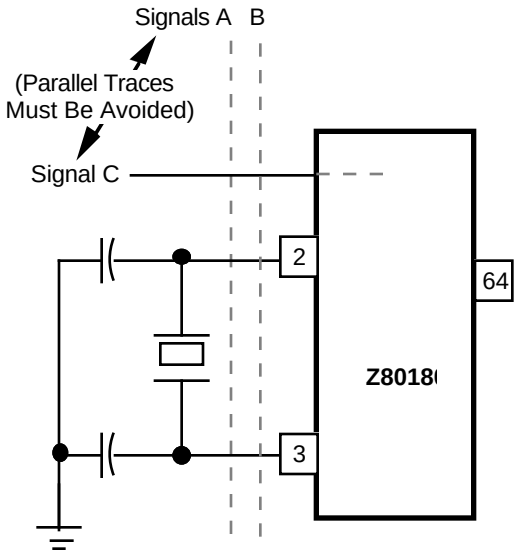
**Start Up Time.** If start up time is excessive, or varies widely from unit to unit, there is probably a gain problem. C1/C2 needs to be reduced; the amplifier gain is not adequate at frequency, or crystal Rs is too large.

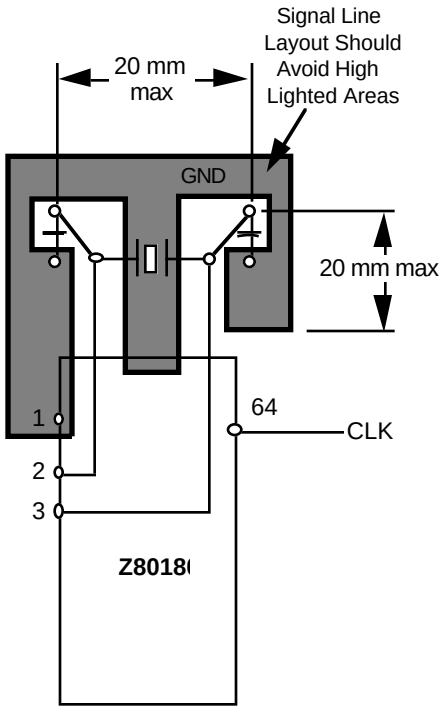
## PRACTICE: CIRCUIT ELEMENT AND LAY OUT CONSIDERATIONS (Continued)

**Output Level.** The signal at the amplifier output should swing from ground to $V_{CC}$. This indicates there is adequate gain in the amplifier. As the oscillator starts up, the signal amplitude grows until clipping occurs, at which point, the loop gain is effectively reduced to unity and constant oscillation is achieved. A signal of less than 2.5 Vp-p is an indication that low gain may be a problem. Either C1/C2 should be made smaller or a low R crystal should be used.




Clock Generator Circuit




Board Design Example
(Top View)

- To prevent induced noice, the crystal and load capacitors should be physically located as close to the LSI as possible.
- Signal lines should not run parallel to the clock oscillator inputs. In particullar, the clock input circuitry and the system clock output (pin 64) should be separated as much as possible.
- $V_{CC}$ power lines should be separated from the clock oscillator input circuitry.
- Resistivity between XTAL or EXTAL and the other pin should be greater than 10 MΩ

**Figure 9. Circuit Board Design Rules**

## SUMMARY

Understanding the Theory of Operation of oscillators, combined with practical applications, should give designers enough information to design reliable oscillator circuits. Proper selection of crystals and load capacitors, along with good layout practices, results in a cost effective, trouble free design.Reference the following text for Zilog products with on-chip oscillators and their general/ specific requirements.

## ZILOG PRODUCT USING ON-CHIP OSCILLATORS

Zilog products that have on-chip oscillators:

Z8® Family: All
Z80®: C01, C11, C13, C15, C50, C90, 180, 181, 280
Z8000®: 8581
Communications Products: SCC™, ISCC™, ESCC™

## ZILOG CHIP PARAMETERS

The following are some recommendations on values/parameters of components for use with Zilog on-chip oscillators. These are only recommendations; no guarantees are made by performance of components outside of Zilog ICs. Finally, the values/parameters chosen depend on the application. This App Note is meant as a guideline to making these decisions. Selection of optimal components is always a function of desired cost/performance tradeoffs.

**Note:** All load capacitance specs include stray capacitance.

### Z8 Family

General Requirements:

Crystal Cut: AT cut, parallel resonant, fundamental mode
Crystal Co: < 7 pF for all frequencies.
Crystal Rs: < 100 ohms for all frequencies.
Load Capacitance: 10 to 22 pf, 15 pF typical.

Specific Requirements:

8604: xtal or ceramic, f = 1 - 8 MHz.
8600/10: f = 8 MHz.
8601/03/11/13: f = 12.5 MHz.
8602: xtal or ceramic, f = 4 MHz.
8680/81/82/84/91: f = 8, 12, 16, MHz.
8671: f = 8 MHz.
8612: f = 12, 16 MHz.
86C08/E08: f = 8, 12 MHz.
86C09/19: xtal/resonator, f = 8 MHz, C = 47 pf max.
86C00/10/20/30: f = 8, 12, 16 MHz
86C11/21/91/40/90: f = 12, 16, 20 MHz.
86C27/97: f = 4, 8 MHz.
86C12: f = 12, 16 MHz.
Super8 (all): f = 1 - 20 MHz.

### Z8000 Family (8581 only)

General Requirements:

Crystal cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pF for all frequencies.
Crystal Rs: < 150 ohms for all frequencies.
Load capacitance: 10 to 33 pF.

### Z80 Family

General Requirements:

Crystal cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pF for all frequencies.
Crystal Rs: < 60 ohms for all frequencies.
Load capacitance: 10 to 22 pF.

Specific Requirements:

84C01: C1 = 22 pF, C2 = 33 pF (typ); f = DC to 10 MHz.
84C90: DC to 8 MHz.
84C50: same as 84C01.
84C11/13/15: C1 = C2 = 20 -33 pf; f = 6 -10 MHz
80180: f = 12, 16, 20 MHz (Fxtal = 2 x sys. clock).
80280: f = 20 MHz (Fxtal = 2 x Fsysclk).
80181: TBD.

### Communications Family

General Requirements:

Crystal cut: AT cut, parallel resonant, fundamental mode.
Crystal Co: < 7 pF for all frequencies.
Crystal Rs: < 150 ohms for all frequencies.
Load capacitance: 20 to 33 pF.
Frequency: cannot exceed PCLK.

Specific Requirements:

8530/85C30/SCC: f = 1 - 6 MHz (10 MHz SCC), 1 - 8.5 MHz (8 MHz SCC).
85130/ESCC (16/20 MHz), f = 1 - 16.384 MHz.
16C35/ISCC: f = 1 -10 MHz.

## REFERENCES MATERIALS AND ACKNOWLEDGEMENTS

Intel Corp., Application Note AP-155, “Oscillators for Micro Controllers”, order #230659-001, by Tom Williamson, Dec. 1986.

Motorola 68HC11 Reference Manual.

National Semiconductor Corp., App Notes 326 and 400.

Zilog, Inc., Steve German; Figures 4 and 8.

Zilog, Inc., Application Note, “Design Considerations Using Quartz Crystals with Zilog Components” - Oct. 1988.

Data Sheets; CTS Corp. Knights Div., Crystal Oscillators.

# INTERFACING THE ISCC™ TO THE 68000 AND 8086

## INTRODUCTION

The ISCC™ uses its flexible bus to interface with a variety of microprocessors and microcontrollers; included are the 68000 and 8086.

The Z16C35 ISCC is a Superintegration form of the 85C30/80C30 Serial Communications Controller (SCC). Super integration includes four DMA channels, one for each receiver and transmitter and a flexible Bus Interface Unit (BIU). The BIU supports a wide variety of buses including the bus types of the 680X0 and the 8086 families of microprocessors.

This Application Note presents the details of BIU operation for both slave peripheral and DMA modes. Included are application examples of interconnecting an ISCC to a 68000 and a 8086 (These examples are currently under test).

## ISCC BUS INTERFACE UNIT (BIU)

The following subsections describe and illustrate the functions and parameters of the ISCC Bus Interface Unit.

### Overview

The ISCC™ contains a flexible bus interface that is directly compatible with a variety of microprocessors and microcontrollers. The bus interface unit adds to the chip by allowing ease of connection to several standard bus configurations; among others are the 68000 and the 8086 family microprocessors. This compatibility is achieved by initializing the ISCC after a reset to the desired bus configuration.

The device also configures to work with a variety of other 8- or 16-bit bus systems and is used with address/data multiplexed or non-multiplexed buses. In addition, the wait/ready handshake, the interrupt acknowledge, and the bus high byte/low byte selection are all programmable. Separate read/write, data strobe, write, read, and address strobe signals are available for direct system interface with a minimum of external logic.

### Modes Description

There are basically two bus modes of operation: multiplexed and non-multiplexed. In the multiplexed bus mode, the ISCC internal registers are directly accessible as separate registers with their own unique hardware addresses. By contrast, in the non-multiplexed mode, all registers access through an internal pointer which first loads with the register address. Loading of the pointer is done as a data write. In either case, there are some external addressing signals.

Chip Enable (CE) allows external selection through the decode of upper order address bits like accessing separate chips. A separate input (not part of the AD15-AD0 bus connection) selects between the internal SCC and DMA sections of the chip. This input is A0/SCC/DMA and provides direct transfers to the appropriate chip subsystem; either multiplexed or non-multiplexed bus mode.

A second separate input (not part of the AD15-AD0 bus connection) provides for a selection between the internal SCC; both channels A and B (Table A-1). This input is A1/A/B and provides direct transfers to the appropriate SCC channel when A0/SCC/DMA selects the SCC; either multiplexed or non-multiplexed bus mode. Note that these two signals, A1/A/B and A0/SCC/DMA, are inputs when

## ISCC BUS INTERFACE UNIT (BIU) (Continued)

the ISCC is a slave peripheral; they become outputs when the ISCC is a bus master during DMA operations.

**Table 1. Accessing the ISCC Registers**

| A0/SCC/DMA | A1/A/B | ACCESS |
|---|---|---|
| 1 | 1 | SCC Channel A |
| 1 | 0 | SCC Channel B |
| 0 | x | DMA |

The following discussions assume knowledge of the SCC Serial Communications Controller operations and refer to internal register designations. For a detailed explanation, refer to the SCC Technical Manual.

### Non-Multiplexed Bus Operation

When the ISCC initializes for non-multiplexed operation, Write Register 0 (WR0) takes on the form of WR0 in the Z8530, Write Register Bit Functions (Figure A-1). Register addressing for the SCC section is (except for WR0 and RR0) accomplished as follows. Programming the write registers requires two write operations. Reading the read registers requires both a write and a read operation.

The first write is to WR0 which contains three bits that point to the selected register (note the point high command). The second write is the actual control word for the selected register. If the second operation is a read, the selected register is accessed. When in the non-multiplexed mode, all registers in the SCC section of the ISCC, including the data registers, access this way.

The pointer register automatically clears after the second read or write operation so WR0 (or RR0) addresses again. There is no direct access to the data registers. They are addressed through the pointer (this is in contrast to the Z8530 which allows direct addressing of the data registers through the C/D pin).

When the ISCC starts for non-multiplexed operation, register addressing for the DMA section is (except for CSAR) accomplished as follows. It is completely independent of the SCC section register addressing. Programming the write registers requires two write operations and reading the read registers requires both a write and a read operation. The first write is to the Command Status Address Register (CSAR) which contains five bits that point to the selected register (CSAR bits 4-0). The second write is the actual control word for the selected register. If the second operation is a read, the selected register is accessed. The pointer bits automatically clear after the second read or write operation so CSAR addresses again. When in the non-multiplexed mode, all registers in the DMA section of the ISCC are accessed.

### Multiplexed Bus Operation

When the ISCC initializes for multiplexed bus operation, all registers in the SCC section are directly addressable with the register address occupying AD5 through AD1 or AD4 through AD0 (Shift Left/Shift Right modes).

The Shift Left/Shift Right modes for the address decoding of the internal registers (multiplexed bus) are separately programmable for the SCC and DMA sections. For the SCC section, the programming and operation is the same as the SCC; programming occurs through Write Register 0 (WR0), bits 1 and 0 , and Write Register Bit Functions (Figure A-2). The programming of the Shift Left/Shift Right modes for the DMA section occurs in the BCR, bit 0. In this case, the shift function is similar to the SCC section; with Left Shift, the internal register addresses decode from bits AD5 through AD1. In Right Shift, the internal register addresses decode from bits AD4 through AD0.

During multiplexed bus mode selection, Write Register 0 (WR0) becomes WR0 in the Z8030, Write Register Bit Functions (Figure A-2).

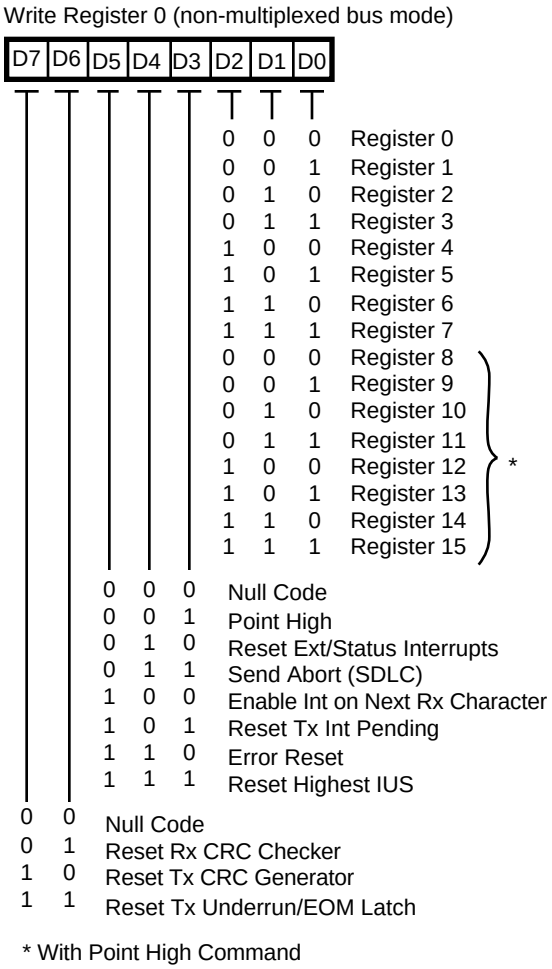



**Figure 1. Write Register 0 Bit Functions (Non-Multiplexed Bus Mode)**

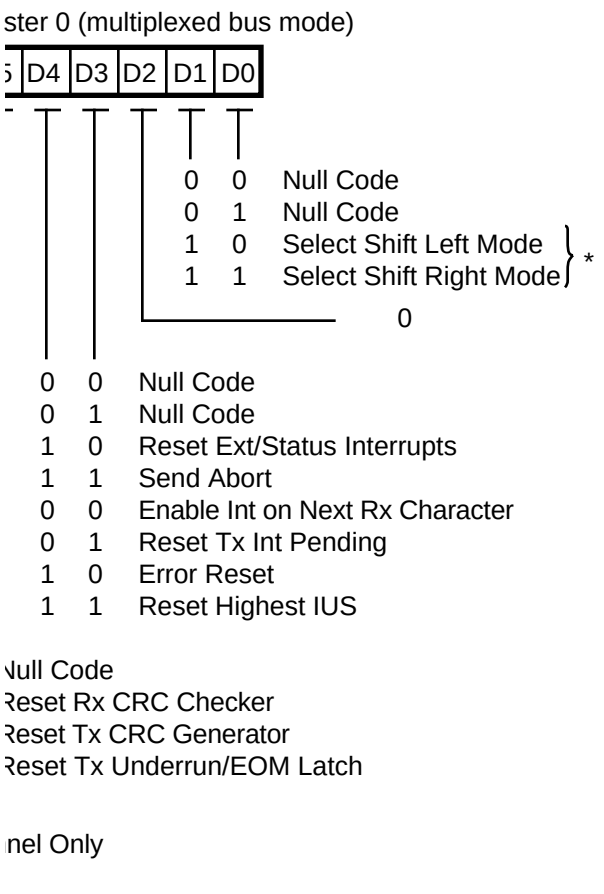



**Figure 2. Write Register 0 Bit Functions (Multiplexed Bus Mode)**

## BUS DATA TRANSFERS

All data transfers to and from the ISCC™ are done in bytes regardless of whether data occupies the lower or upper byte of the 16-bit bus. Bus transfers as a slave peripheral are done differently from bus transfers when the ISCC is the bus master during DMA transactions. The ISCC is fundamentally an 8-bit peripheral but supports 16-bit buses in the DMA mode. Slave peripheral and DMA transactions appear in the next sections.

### Data Bus Transfers as a Slave Peripheral

When accessed as a peripheral device (when the ISCC is not a bus master performing DMA transfers), only 8 bits transfer. During ISCC register read, the byte data present on the lower 8 bits of the bus is replicated on the upper 8 bits of the bus. Data is accepted by the ISCC only on the lower 8 bits of the bus.

### ISCC™ DMA Bus Transfers

During DMA transfers, when the ISCC is bus master, only byte data transfers occur. However, data transfers to or from the ISCC on the upper 8 bits of the bus or on the lower 8 bits of the bus. Moreover, odd or even byte transfers activate on the lower or upper 8 bits of the bus. This is programmable and explained next.

During DMA transfers to memory from the ISCC, only byte data transfers occur. Data appears on the lower 8 bits and replicates on the upper 8 bits of the bus. Thus, the data is written to an odd or even byte of the system memory by address decoding and strobe generation.

During DMA transfers to the ISCC from memory, byte data only transfers. Normally, data appears only on the lower 8 bits of the bus. However, the byte swapping feature

## BUS DATA TRANSFERS (Continued)

determines which byte of the bus data is accepted. The byte swapping feature activates by programming the Byte Swap Enable bit to a 1 in the BCR. The odd/even byte transfer selection occurs by programming the Byte Swap Select bit in the BCR. If Byte Swap Select is a 1, then even address bytes (transfers where the DMA address has A0 = 0) are accepted on the lower 8 bits of the bus. Odd address bytes (transfers where the DMA address has A0 = 1) are accepted on the upper 8 bits of the bus. If Byte Swap Select is a 0, then even address bytes (transfers where the DMA address has A0 = 0) are accepted on the upper 8 bits of the bus. Odd address bytes (transfers where the DMA address has A0 =1) are accepted on the lower 8 bits of the bus.

### Bus Interface Handshaking

The ISCC™ supports data transfers by either a data strobe (DS) combined with a read/write (R/W) status line, or separate read (RD) and write (WR) strobes. These transactions activate via chip enable (CE).

ISCC programming generates interrupts upon the occurrence of certain internal events. The ISCC internally prioritizes its own interrupts, therefore, the ISCC presents one interrupt to the processor even though lower priority internal interrupts may be pending. Interrupts are individually enabled or disabled. Refer to the sections on the SCC core.

Interrupt Acknowledge (INTACK) is an input to the ISCC showing that an interrupt acknowledge cycle is progressing. INTACK is programmed to accept a status acknowledge, a single pulse acknowledge, or a double pulse acknowledge. This programming activates in the BCR. The double pulse acknowledge is compatible with 8X86 family microprocessors and the status acknowledge is compatible with 68000 family microprocessors.

During an interrupt acknowledge cycle, the SCC and DMA interrupt priority daisy chain internally resolves. Thus, the highest priority internal interrupt is presented to the CPU.

The ISCC can return an interrupt vector that encodes with the type of interrupt pending enabled during this acknowledge cycle. The ISCC may request an interrupt but not return an interrupt vector [note that the no vector bit(s) in the SCC section (WR9 bit 1) and in the DMA section (ICR bit 5) individually control whether or not an interrupt vector returns by these cores]. The interrupt vector can program to include a status field showing the internal ISCC source of the interrupt. During the interrupt acknowledge cycle, the ISCC returns the interrupt vector when INTACK, RD or DS go active and IEI is high (if the ISCC is not programmed for the no vector option).

During the programmed pulsed acknowledge type (whether single or double), INTACK is the strobe for the interrupt vector. Thus when INTACK goes active, the ISCC drives the bus and presents the interrupt vector to the CPU. When the status acknowledge type programs, the ISCC drives the bus with the interrupt vector when RD or DS are active.

WAITRDY programs to function either as a WAIT signal or a READY signal using the BCR write. When programmed as a wait signal, it supports the READY function of 8X86 family microprocessors. When programmed as a ready signal, it supports the DTACK function of 680x0 family microprocessors.

The WAIT/RDY signal functions as an output when the ISCC is not a bus master. In this case, this signal serves to indicate when the data is available during a read cycle, when the device is ready to receive data during a write cycle, and when a valid vector is available during an interrupt acknowledge cycle.

When the ISCC is the bus master (DMA section has taken control of the bus), the WAIT/RDY signal functions as a WAIT or RDY input. Slow memories and peripheral devices use WAIT to extend the data strobe (/DS) during bus transfers. Similarly, memories and peripheral devices use RDY to indicate valid output or that it is ready to latch input data.

## CONFIGURING THE BUS

The bus configuration programming is done in two separate steps (actually it is one operation), to enable the write to the Bus Configuration Register (BCR). The first operation that accesses the ISCC after a device reset must be a write to the BCR since this is the only time that the BCR is accessible. Before and during the write, various external signals are sampled to program bus configuration parameters. During this write, the AØ/SCC//DMA pin must be Low.

Address strobe programs multiplexed/non-multiplexed selection. In a non-multiplexed bus environment, address strobe (as an input) is not used but tied high through a suitable pull-up resistor. Thus, no address strobe is present before the BCR write. Then, when write to the BCR takes place, the non-multiplexed mode is programmed because there is no address strobe before this first write to the device. Note that address strobe becomes an output during DMA operations so it is not tied directly to $V_{CC.}$

During the write operation to the BCR, the A1/A/B input is sampled to select the function of the WAIT/RDY pin (Table A-2). When the BCR Write is to the SCC Channel A (A1/A//B High during the BCR write), the WAIT/RDY signal functions as a wait. When the BCR Write is to Channel B (A1/A//B Low during the BCR write), the WAIT/RDY signal functions as a ready.

**Table 40. Signals Sampled During the BCR Write**

| A1/A//B | WAIT/RDY Function |
|---|---|
| 1 | WAIT (8086 RDY compatible) |
| 0 | READY (68000 DTACK compatible) |

This programming affects the function of the WAIT/RDY signal both as an input, when the ISCC is bus master during DMA operations, and as an output when the ISCC is a bus slave.

With this programming, the ISCC is immediately configured to function successfully on this first and subsequent bus transactions. The remaining bus configuration options are programmed by the value written to the BCR.

Bit 0 of the BCR controls the Shift Left/Shift Right address decoding modes for the DMA section. In this case, the shift function is similar to the SCC section. During Left Shift, the internal register addresses decode from bits AD5 through AD1. During Right Shift, the internal register addresses are decode from bits AD4 through AD0. This function is only applicable in the multiplexed bus mode.

Bits 1 and 2 of the BCR control the interrupt acknowledge type as shown in the Table A-3.

**Table 41.** BCR Control of Interrupt Acknowledge

| BCR bit 2 | BCR bit 1 | Interrupt Acknowledge |
|---|---|---|
| 0 | 0 | Status Acknowledge |
| 0 | 1 | Pulsed Acknowledge (single) |
| 0 | 1 | Reserved (action not defined) |
| 1 | 1 | Double Pulsed Acknowledge |

The Status Acknowledge remains active throughout the interrupt cycle and is directly compatible with the 680x0 family interrupt handshaking. The Status Acknowledge signal latches with the rising edge of AS for multiplexed bus operation. It latches by the falling edge of the strobe (RD or DS) for non-multiplexed bus operation. The Pulsed Acknowledges are timed to be active during a specified period in the interrupt cycle. The Double Pulsed Acknowledge is directly compatible with the 8x86 family interrupt handshaking. Refer to the timing diagrams in the ISCC Product Specification for details on the Acknowledge signal operation.

Reserve bits 3, 4, and 5 of the BCR program as zeros. Bits 6 and 7 of the BCR control the byte swap feature (Table A-4). Byte swap is applicable only in DMA transfers when the ISCC is the bus master and only affects ISCC data acceptance (transfers from memory to the ISCC).I

**Table 42. Byte Swap Contro**

| Enable (BCR bit 7) | DMA Data Read by the ISCC |
|---|---|
| 0 | lower 8 bits of bus only |
| 1 | upper or lower 8 bits of bus |

| Swap Select* | A0 | DMA Data read by the ISCC |
|---|---|---|
| 0 | 0 | upper 8 bits of bus |
| 0 | 1 | lower 8 bits of bus |
| 1 | 0 | lower 8 bits of bus |
| 1 | 1 | upper 8 bits of bus |

* BCR bit 6

## APPLICATIONS EXAMPLES

The following application examples explain and illustrate the methods of interfacing the ISCC to a Motorola 68000 and an Intel 8086.

### 68000 Interface to the ISCC

Figure A-3 shows a connection of the ISCC to a 68000 microprocessor. The 68000 data bus connects directly, or through bus transceivers, to the ISCC address/data bus. R/W and RESET also directly connect. In this example, the ISCC is on the lower half of the bus; DS of the ISCC connects to LDS of the 68000. The processor address lines decode to produce a chip enable for the ISCC. In addition, processor addresses A1 and A2 connect to A0/SCC/DMA and A1/A/B, respectively, through a tri-state driver.

The driver is normally ON (enabled) but turns OFF by BGACK to grant the bus to ISCC for DMA transfers. This is done since the A0/SCC/DMA and A1/A/B pins become outputs during DMA transfers and should not drive the system address bus. RD and WR tie high through independent pull-ups. They are not used in this application but become active outputs during DMA transfers and are not tied directly to $V_{CC}$.

Although not shown in Table A-5, the A0/SCC/DMA and A1/A/B pins may be decoded during DMA transfers to identify the active DMA channel.

**Table 43. DMA A/B Channel Decode**

| A1/A/B | A0/SCC/DMA | DMA Channel |
|---|---|---|
| 1 | 1 | Receiver Channel A |
| 1 | 0 | Transmitter Channel A |
| 0 | 1 | Receiver Channel B |
| 0 | 0 | Transmitter Channel B |

External logic can use this information to abort a DMA in progress.

For normal slave device bus interaction, a DTACK is generated. WAIT/RDY is programed for ready operation and INTACK programs for the status type. WAIT/RDY generates a DTACK for normal data transfers and interrupt responses. Additional logic may be required when other interrupt sources are present.

During DMA transfers, the ISCC becomes bus master. Becoming bus master is done through the BUSREQ output and BUSACK input signals of the ISCC. They connect to an external bus arbitration circuit. This circuit performs bus arbitration for multiple bus master requests and generates bus grant acknowledge (BGACK) which controls certain bus drive signal sources.

When the ISCC becomes the bus master, a 32-bit address generation by the DMA section is output on the ISCC address/data bus. The lower 16 bits of this address store in an external latch by AS (Address Strobe). Also, the upper 16 bits of this address store in an external latch by UAS (Upper Address Strobe). With BGACK low (active) and with the processor address lines tri-stated, the latch outputs drive the system address bus.

AS is pulled high by an external resistor. This pull-up insures an inactive AS (at a logic high level) when the ISCC is not driving this signal. Therefore, on power up or after a RESET, AS is inactive and programs the non-multiplexed bus mode on BCR write.

In this application, the outputs of the address latches are connected to the address bus so that A1 through A23 of the ISCC drives the system address bus (the ISCC provides a total of 32 address lines). A0 from the address latch is diverted to logic which generates UDS and LDS bus signals from the ISCC data strobe (DS). UDS is generated when A0 is low and LDS is generated when A0 is high. The lower and upper data strobes are applied to the system bus through tri-state drivers which are enabled only when BGACK is active. Bus direction is now controlled by the ISCC R/W signal which is now an output.

For initialization, the BCR write (the first write to the ISCC after RESET) is done with A2 = 0 (A1/A/B ISCC input at logic low). This selects the ready option of the WAIT/RDY signal to conform to the 68000 bus style. The AS signal programming of the non-multiplexed bus has already been discussed. The BCR is written with C0H to enable byte swapping. It also selects the sense of byte swapping with respect to A0 appropriate to this bus style and selects the STATUS type of interrupt acknowledge.

### 8086 Interface with the ISCC

Figure A-4 shows the connection of the ISCC to an 8086 microprocessor and companion clock state generator. In this application, the ISCC connects for multiplexed address access to the internal ISCC registers. AD15 through AD0 of the 8086 connect directly, or through a bus transceiver, to the corresponding AD15 through AD0 address/data ISCC bus pins. RD and WR are directly compatible and tie together to form the read and write bus signals.

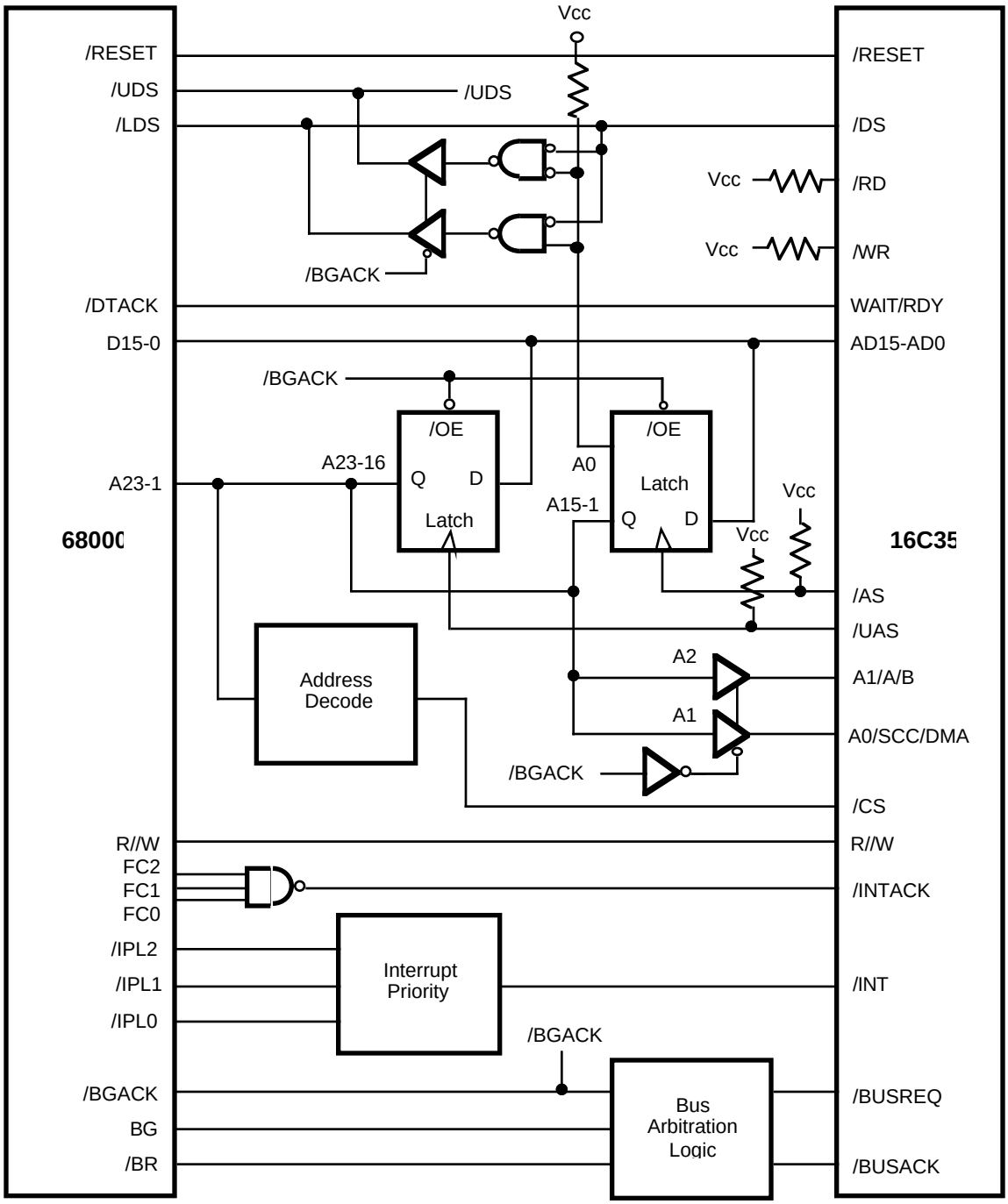
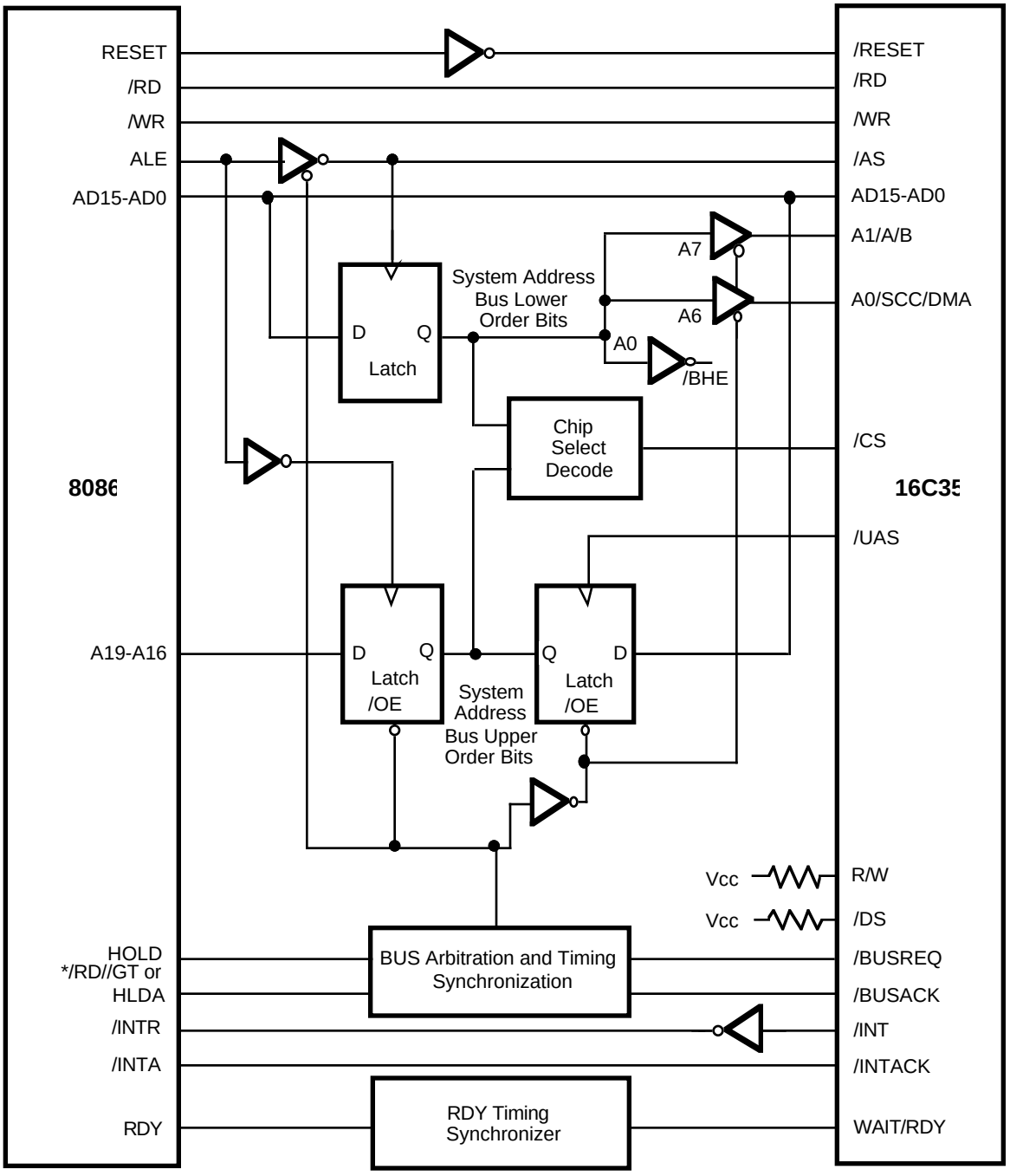



**Figure 3. ISCC Interface to a 68000 Microprocessor**

## APPLICATIONS EXAMPLES (Continued)




* maximum mode

**Figure 4. ISCC Interface to an Intel 8086 Microprocessor**

When the ISCC becomes a bus master during DMA operations, RD and WR of the 8086 are tri-stated which allows the corresponding ISCC signals to control the bus transactions. The sense of RESET reverses, so the ISCC RESET signal inverts from the reset applied to the 8086 from the clock state generator.

RD/WR and DS of the ISCC are inactive in this application and tie high. They tie high through independent pull-ups since these signals become active when the ISCC is bus master during DMA transactions.

Assuming other devices in the system, the ISCC chip enable input (CE) activates from a decode of the address. In this example, the ISCC internally decodes addresses A1 through A5 and uses A6 and A7, externally. Thus, the address decode circuitry decodes address lines A0 and A8 and above. The decode of A0 for chip enable places the ISCC as an 8-bit peripheral on the lower byte of the bus. A0 and the upper level address lines (including A6 and A7) demultiplex from the 8086 address/data bus through a latch strobed by ALE.

The demultiplexed addresses A6 and A7 connect to A0/SCC/DMA and A1/A/B, respectively, of the ISCC to control selection of the DMA and SCC channels A and B. This connects through the tri-state drivers. They enable when the 8086 is the bus master and disable when the ISCC is bus master. This prevents the ISCC from improperly driving the system address bus since A0/SCC/DMA and A1/A/B become active outputs when the ISCC is the bus master.

The address map for the ISCC appears in Table A-6 for this application.

**Table 44. ISCC Address Map**

| A0 | A1-A5 | A6 | A7 | Registers Addressed |
|---|---|---|---|---|
| 1 | x | x | x | ISCC not enabled |
| 0 | - | 0 | x | DMA Registers per A1 - A5 |
| 0 | - | 1 | 1 | SCC Core Channel A Registers |
| 0 | - | 1 | 0 | SCC Core Channel B Registers |

Since A0 specifies the lower byte of the bus and includes the chip enable decode, the internal ISCC register addresses decode without A0. Thus, Table 6 implies that the Left Shift address decode selection is made for both the SCC and DMA sections of the ISCC. The left shift selection is the default selection after reset. Left/Right Shift selection programming is discussed later.

The ALE signal of the 8086 applies to AS of the ISCC through an inverting tri-state buffer. The buffer disables when the ISCC becomes a bus master during DMA transactions. This prevents conflicts since ALE remains active even when the 8086 is in the HOLD mode during DMA transfers. Now, the ISCC AS is an active output. The address strobe for the demultiplexing latch of addresses A0 through A15 connects on the ISCC side of the ALE tri-state buffer. This allows the latch to serve two functions; to hold either the 8086 or the ISCC address when it is bus master.

After reset, ALE is active and the tri-state buffer enabled. This supplies address strobes to the ISCC. The presence of one of these address strobes, before writing to the BCR, programs the ISCC to the multiplexed bus mode of operation. The ISCC chip enable (CE) can be inactive and still recognize an address strobe (AS) before the BCR write (Figure 4 shows open latches when the input strobe is low).

When the ISCC is bus master during DMA transactions, BHE generates from A0. This is done from the output of the lower order address latch through an inverting tri-state driver. This driver enables only when the ISCC is the bus master. Whole word transfers are not done by the ISCC DMA, thus, BHE generated for the ISCC is always the inverse of A0.

The upper bus system address lines demultiplex from the 8086 and the ISCC in separate latches. Like the 68000 example, high order address lines from the ISCC latch via UAS (upper address strobe). The separate latches drive the same upper order address lines. A16 from the ISCC connects to the corresponding A16 address bus line as derived from the 8086. The output of the two latches alternately enable depending upon bus mastership.

The diagram shows INT from the ISCC connected to the 8086 INTR input via an inverter since these signals are of opposite sense. In actual practice, the ISCC interrupt request is first processed by an interrupt priority circuit. INTA (Interrupt Acknowledge) of the 8086 connects directly to the INTACK input of the ISCC. Conforming to the 8086 style of interrupt acknowledge, the ISCC is programed to the Double Pulse Interrupt Acknowledge type. When this selection occurs, the ISCC responds to two interrupt acknowledge pulses. The first pulse is recognized but no action follows. The second pulse causes the ISCC to go active on the data bus and return the interrupt vector to the CPU. This action also takes place with the Single Pulse Interrupt Acknowledge type selection, except that the bus goes active with the first and only interrupt acknowledge pulse.

To start, the BCR write (first write to the ISCC after RESET) is done with A7 = 1 (A1/A/B ISCC input at logic high). This selects the wait option of the WAIT/RDY signal to conform to the 8086 bus style. The AS signal programming of the multiplexed bus was covered earlier. The BCR is written with 86H to enable byte swapping, select the sense of the byte swapping with respect to A0 (appropriate to this bus style), and select the Double Pulse type of interrupt acknowledge.

When the ISCC™ begins DMA transfers, it communicates requests for the bus through BUSREQ and BUSACK. The 8086 receives and grants bus requests through HOLD and HLDA in the minimum mode and through RQ/GT in the maximum mode. Depending upon the system requirements, there could be more than one potential bus master. Therefore, there is a requirement for a bus arbitration circuit.

The minimum mode connection is relatively straightforward. The maximum mode configuration requires a translation of the ISCC BUSREQ and BUSACK signals into/from the 8086 RQ/GT timed pulse style of handshake. Refer to the information on the 8086 for detailed application information.

The ISCC™ WAIT/RDY output is compatible with the 8086 clock generator RDY input except that one edge of the signal must be synchronous with the 8086 clock. The synchronization occurs through external circuitry. Refer to the information on the 8086 for detailed application information.

# ZILOG SCC
## Z8030/Z8530
## QUESTIONS AND ANSWERS

*This document addresses the most commonly asked questions about Zilog's SCC. These questions fall into the following five categories:*

- Hard ware Considerations
- Interrupt s and Polling
- Asychronous mode
- Sychronous Mode
- Miscellaneous Questions

## HARDWARE CONSIDERATIONS

This section includes questions and answers on the hardware interface, the clocks, the FIFO, special modes (Local Loopback, DPLL, Manchester), and internal timing consideration.

### Hardware (Includes DMA Interface)

***Q. What is the SCC transistor count?***

A. Approximately 6000 gates, or 18,000 transistors.

***Q. What is the difference between the Z8030 and the Z8530?***

A. The Z8030 and Z8530 are packaged from the same die. The multiplexed bus (Z8030) or non-multiplexed bus (Z8530) version of the chip is selected at packaging time by an internal bonding option.

***Q. Can /AS be active only when the Z8030 is being accessed and High all other times?***

A. Since the interrupt pending bits (IPs) are updated on address strobes, interrupts will not occur unless /AS is continuous.

***Q. How do /WR and /CE interact on the Z8530?***

A. /WR and /CE are ANDed to enable a transparent latch. Data is latched on the falling edge when both /CE and /WR go Low.

***Q. How many register pointers does the Z8530 have?***

A. The SCC has only one register pointer for both channels. The SIO (Z844X) has two, one for each channel.

***Q. Do you have to write to the pointer with the Z8530 to access WR0 or RR0?***

A. No. Both registers are accessed automatically without first writing to the pointer.

***Q. Does /CE (/CS) have to be High during an interrupt acknowledge cycle?***

A. No.

***Q. Does the SCC support full duplex DMA?***

A. The SCC allows full duplex DMA transfers by using the DTR/REQ and W/REQ as two separate DMA control lines for transmit request and receive request on each channel.

***Q. When using full duplex DMA, how do you program W/REQ?***

A. W/REQ should be programmed for receive and DTR/REQ pin should be programmed for transmit.

***Q. Can both channels make simultaneous DMA requests?***

A. Yes.

***Q. Do you have to reset the SCC in hardware?***

A. No. A software reset is the same as a hardware reset, (WR9 CO). It also does not matter whether the Z8030 is in shift right or shift left mode because the address is the same in either.

## HARDWARE CONSIDERATIONS (Continued)

***Q. Do you need to clear the reset bit in WR0 after a software reset?***

A. The reset is clocked with PCLK; so it must be active during reset.

***Q. How long after a hardware reset should you wait before programming the SCC.***

A. Four PCLKs.

***Q. Why does the SCC initialization require that the External Status Interrupts be reset twice?***

A. Because of the possibility of noise causing an interrupt pending bit (IP) to be set. The second reset guarantees that the latch is clear. If the latch is closed high and the external signal is low, the first reset will open the latch at the high-to-low transition causing an interrupt.

## Clocks

***Q. Does PCLK have to have a 50% duty cycle?***

A. The duty cycle doesn't have to be 50% as long as the minimum specification is met.

***Q. Can the SCC PCLK be stretched?***

A. Yes, as long as the pertinent specification is met. However, this could cause a problem if PCLK is used to generate the bit rate.

***Q. The bit rate generator is driven from what sources?***

A. It may be driven from the RTxC pin or PCLK, or from a crystal.

***Q. How do you connect a bit rate crystal to the SCC?***

A. A crystal can be connected between RTxC and SYNC to supply the clock if the SCC is programmed for WR11 D7-1.

***Q. What is the crystal specification?***

A. It is a fundamental, parallel resonant crystal. For further details see the "Design Considerations Using Quartz Crystals with Zilog's Components" Application Note.

***Q. Can RTxC on both channels be driven from the same crystal.***

A. No. A separate crystal should be used for each channel. The crystal should be connected between /SYNC and RTxC of the respective channels. The alternate solution may be to use crystal on one channel and reflect the clock out of the TRxC output and feed it into another channel.

***Q. How do you select a crystal frequency?***

A. Time constant: (Clock Frequency/2 x Bit rate x clock factor) - 2. Two examples are given below:

| For PCLK = 3.6864 MHz | | | For PCLK = 3.9936 MHz | | |
|---|---|---|---|---|---|
| **Bit Rate** | **TC** | **Error** | **Bit Rate** | **TC** | **Error** |
| 38400 | 46 | - | 19200 | 102 | - |
| 19200 | 94 | - | 9600 | 206 | - |
| 9600 | 190 | - | 7200 | 275 | 12% |
| 7200 | 254 | - | 4800 | 414 | - |
| 4800 | 382 | - | 3600 | 553 | .06% |
| 3600 | 510 | - | 2400 | 830 | - |
| 2400 | 766 | - | 2000 | 996 | .04% |
| 1200 | 1534 | - | 1800 | 1107 | .03% |
| | | | 1200 | 1662 | - |
| | | | 600 | 3326 | - |
| | | | 300 | 6654 | - |
| | | | 150 | 13310 | - |
| | | | 134.5 | 14844 | .0007% |
| | | | 110 | 18151 | .0015% |
| | | | 75 | 26622 | - |
| | | | 50 | 39934 | - |

***Q. Why are there different Clock factors?***

A. These clock factors enable the SCC to sample the center of the data cell. In the 16x mode, the SCC divides the bit cell into 16 counts and samples on count 8. Clock factors are generally only used with Asynchronous modes.

***Q. How is the error in the receive/transmit clock reduced?***

A. The ideal way to reduce this error is by adjusting the crystal frequency such that only an integer value of TC is yielded when the equation is used.

***Q. What are the maximum transfer rates?***

A. The following table shows the PCLK rates (in bps).

| | 4 MHz | 6 MHz | 8 MHz | 10 MHz | 16 MHz | 20 MHz |
|---|---|---|---|---|---|---|
| **Asynchronous mode:** | | | | | | |
| External clock | | | | | | |
| 6x mode (no BRG) | 250K | 375K | 500K | 635K | 1M | 1.25M |
| BRG | | | | | | |
| 16x mode (TX + 0) | 62.5K | 93.75K | 125K | 156.5K | 250K | 312.5K |
| **Synchronous mode:** | | | | | | |
| Using external clock | 1M | 1.5M | 2M | 2.5M | 4M | 5M |
| Using DPLL, FM encoding | 250K | 375K | 500K | 625K | 1M | 1.25M |
| Using DPLL, MRZ/NRZI encoding | 125K | 187.5K | 250K | 312.5K | 500K | 625K |
| Using DPLL, FM, BRG | 62.5K | 93.75K | 125K | 156.25K | 250K | 312.5K |
| Using DPLL, NRZ/NRZI, BRG | 32.25K | 46.88K | 62.5K | 78.125K | 125K | 156.25K |

***Q. Can the maximum transfer rate using an external clock be achieved?***

A. Yes, but it is not trivial. In order to achieve the maximum rate on transmit, the SCC should have a dedicated processor or DMA. For example, at a 1 MHz rate, a byte must be loaded into the SCC every 8 microseconds. To achieve the maximum rate on receive, requires that the receive clock and the SCC PCLK be synchronized. (RTxC to PCLK setup time at maximum rate in the Product Specification.) It is probably easier to use a slightly faster PCLK SCC, or back off slightly from the maximum rate.

## FIFO

***Q. How do you avoid an overrun in the received FIFO?***

A. The receive buffer must be read before the recently received data character on the serial input is shifted into the receive data FIFO. This FIFO is three bytes deep. Thus, if the buffer is not read, the fifth character just arrived causes an overrun condition. There is no bit that can be set or reset to disable the buffering.

***Q. What happens when you read an empty FIFO?***

A. You read the last character in the buffer.

***Q. When the FIFO gets locked due to an error condition, can it still receive?***

A. The SCC continues to receive until an overrun occurs.

***Q. Assuming that there are characters available in the FIFO, what happens to them if the receiver goes into the hunt mode?***

A. They will remain in the FIFO until they are either read by the CPU or DMA, or until the channel is reset.

## SPECIAL MODES (LOCAL, LOOPBACK, DPLL, MANCHESTER)

***Q How are the Local, Loopback, and Auto Echo modes implemented?***

A. The TxD and RxD pins are connected through drivers. If both modes are simultaneously enabled, then Auto Echo overrides.

***Q. Can the SCC transmit when the Auto Echo mode is enabled?***

A. No, the transmitter is logically disconnected from the TxD pin.

***Q. Can the Digital Phase Lock Loop (DPLL) be used with NRZ?***

A. The DPLL simply generates the receive clock which is the same for both NRZ and NRZI.

***Q. Do you have to use the DPLL with NRZI and FM encoding?***

A. If the DPLL is not used, a properly phased external clock must be supplied.

***Q. What is the error tolerance for the DPLL?***

A. The DPLL can only tolerate a + or - 1/32 deviation in frequency, or about 3%.

***Q. Can you receive and transmit between two channels on the same SCC using the DPLL to generate both the transmit and receive clocks?***

A. To transmit and receive using the same clock, you need to divide the transmit clock by 16 or 32 to be the same rate for transmitting and receiving, because the DPLL requires a divide-by-16 or -32 on the receiver, depending on the encoding. An external divide-by-16 or -32 is required, and can be connected by outpouring the bit rate generator on the /TRxC pin, through the external divide circuit, and back in the /RTxC pin as an input to the transmitter.

***Q. How fast will Manchester be decoded?***

A. The SCC can decode Manchester data by using the DPLL in the FM mode and programming the receiver for NRZ data. Hence, the 125K bit/s is the maximum rate for decoding at 8MHz SCC. A circuit for encoding Manchester is available from Zilog.

***Q. When will the Time Constant be loaded into the BRG counter?***

A. After a S/W reset or a Zero Count is reached.

***Q. How to run NRZ data using the DPLL?***

A. Use NRZI for DPLL (WR14) but set to NRZ (WR10).

## INTERNAL TIMING

***Q. When does data transfer from the transmit buffer to the shift register?***

A. About 3 PCLK's after the last bit is shifted out.

***Q. How long does it take for a write operation to get to the transmit buffer?***

A. It takes about 5 PCLK's for the data to get to the buffer.

***Q. What is Valid Access Recovery Time?***

A. Since WR/ and RD/ (AS/ and DS/ on the Z8030) have no phase relationship with PCLK, the circuitry generating these internal control signals must provide time for metastable conditions to disappear. This gives rise to a recovery time related to PCLK.

***Q. How long is Valid Access Recovery Time?***

A. On the NMOS SCC, the recovery time is 4 PCLK's, while on the CMOS SCC, the recovery time is 3-3.5 PCLK's.

***Q. Why does the Z8030 require that the PCLK be "at least 90% of the CPU clock frequency for Z8000?"***

A. If the clocks are within 90%, then the setup and hold times will be met. Otherwise, the setup and hold times must be met by the user.

***Q. Does Valid Access Recovery Time apply to all successive accesses to the SCC?***

A. Any access to the SCC requires that the recovery time be observed before a new access. This includes reading several bytes from the receive FIFO, accessing separate bytes on two different channels, etc. When using DMA or block transfer methods, the recovery time must be considered.

***Q. Do the DMA request and wait lines on the SCC take the Valid Access Recovery time into account before they make a request?***

A. No, they are not that intelligent. The user must take this into account, and program the DMA accordingly. For example, by inserting wait states during the memory access between SCC accesses, which will lengthen the time in between SCC accesses, or by requiring the DMA to release the bus between accesses to the SCC, to prevent simultaneous data requests from two channels from violating the recovery time.

***Q. What happens if Valid Access Recovery Time is violated?***

A. Invalid data can result.

***Q. Does Valid Access Recovery Time affect the interrupt acknowledge cycle?***

A. No. The interrupt vector is put on the bus by the SCC during the interrupt acknowledge cycle, but does not require any recovery time.

***Q. Why can some systems violate the recovery time by 1 or 2 PCLK's without affecting the data to the SCC?***

A. This violation may or may not matter to the SCC. This phase relationship between PCLK, /RD, /WR, (/AS, /DS for Z8030) can by ASYNC. The SCC requires some time internally to synchronize these signals. The electrical specs for the SCC indicate a recovery time, which is the worst case maximum.

## INTERRUPT CONSIDERATIONS

***Q. What conditions must exist for the SCC to generate an interrupt request?***

A. Interrupts must be enabled (MIE = 1 and IE = 1). The Interrupt Enable Input (IEI) must be high. The interrupt pending bit (IP) must be set and its interrupt under service bit (IUS) must be reset. No interrupt acknowledge cycle may be active.

***Q. How can the /INTACK signal be synchronized with PCLK?***

A. /INTACK needs to be synchronized with PCLK. This can be accomplished by changing /INTACK only on the falling edge of PCLK by using a D flip-flop that is clocked with the inverted PCLK.

***Q. Is /CE required during an Interrupt Acknowledge cycle?***

A. No.

***Q. How long does /INT stay active low when requesting an interrupt?***

A. If the SCC is operated in a polled mode, the /INT will remain active until the IP bit is reset. For an interrupt acknowledge cycle, the /INT will go inactive shortly after the falling edge of /RD or /DS when the IUS bit is set.

***Q. Can you use the SCC without a hardware interrupt acknowledge?***

A. Yes. If you are not using the hardware daisy chain, you don't need to give an interrupt acknowledge. Tie the intack pin high, enable interrupts, and on responding to an interrupt, check RR3 for the cause, and special receive conditions if you are in receive mode. The internal daisy-chain settling time must still be met. (IEI to IEO delay time specification.)

***Q. How do you acknowledge an interrupt without a hardware interrupt acknowledge?***

A. Reset the responsible interrupt pending bit (IP). The /INT line follows the IP bit.

***Q. When are the IP bits cleared?***

A. A transmitter empty IP is cleared by writing to the data register. A receive character available IP is cleared by reading the data register. The external/status interrupt IP is cleared by the command Reset Ext/Status Interrupts.

***Q. Can the IP bits be set while the SCC is servicing other interrupts?***

A. Yes. If the interrupting condition has a higher priority than the interrupt currently being serviced, it causes another interrupt, thus nesting the interrupt services.

***Q. Can the IUS bits be accessed?***

A. No. They are not accessible.

***Q. When do IUS bits get set?***

A. The IUS bits are set during an interrupt acknowledge cycle on the falling edge or /RD or /DS.

***Q. How do you reset interrupts on the SCC?***

A. The interrupt under service bit (IUS) can be reset by the command "Reset Highest IUS" or 38 Hex to WR0. Reset Highest IUS should be the last command issued in the interrupt service routine.

***Q. Why is the interrupt daisy chain settle time required?***

A. This mechanism allows the peripheral with the highest priority interrupt pending in the hardware interrupt daisy chain to have its interrupt serviced.

***Q. Is there still a settle time if the peripherals are not chained?***

A. Even if only one SCC is used, there still is a minimum daisy-chain settle time due to the internal chain.

***Q. How should the vectors be read when utilizing the /INTACK?***

A. /INTACK should be tied to 5 volts through a register. Erroneous reads can result from a floating INTACK. The interrupt vectors can be read after an interrupt from RR2.

***Q. How is the vector register different from the other registers?***

A. The vector register is shared between both channels. The Write register can be accessed from either channel. Reading "Read Register 2" on Channel A (RR2A) returns the unmodified vector, and RR2B returns the

## INTERRUPT CONSIDERATIONS (Continued)

modified vector that includes status. The vector includes the status bit (VIS, WR9) and determines which vector register is put out on the bus during an interrupt cycle.

***Q. How do you poll the external/status interupt IP bit?***

A. Set the IE bits in WR15 so the conditions are latched and set ext/status master interrupt enable bit in WR1. To guarantee the current status, the processor should issue a Reset External/Status interrupts command in WR0 to open the latches before reading the register. For further details see the SCC Technical Manual, section 3.4.7.

***Q. When should the status in RR1 be checked?***

A. Always read RR1 before reading the data.

***Q. What conditions cause the transmit IP to be set?***

A. Either the buffer is empty, or the flag after CRC is being loaded.

***Q. How do you tell if you have a Zero Count (ZC) interrupt?***

A. This bit is not latched like the other external IP bits. If an external interrupt occurs and none of the other IP bits have changed since the last ext/status interrupt, then the ZC condition caused it. A ZC interrupt will not be generated if there are other ext/status (IP) pending. The ZC stays active for each time only when the count reached zero, approximately two PCLK time periods.

***Q. How do you poll the bits in RR3A?***

A. Enable interrupts in WR1 and disable MIE before polling.

***Q. What happens when the SCC is programmed to interrupt on transmit buffer empty and also to request DMA activity on transmit buffer empty?***

A. This would not be a wise thing to do. The interrupt would occur but the DMA could gain control of the bus and remove the interrupting condition before the interrupt acknowledge could take place. When the CPU recovers control of the bus and starts the interrupt acknowledge cycle, bus confusion results because the peripheral no longer has a reason to interrupt.

***Q. Will IP bit (s) for external status be cleared by the Reset Ext/Status Interrupt?***

A. Yes.

## ASYNCHRONOUS MODE

***Q. Can the Sync Character Load Inhibit function strip characters in Asynchronous mode if not disabled?***

A. Yes. If not disabled it will strip any characters which match the value in the sync character register. Always disable this function in asynchronous mode (WR3, bit D1).

***Q. What controls the DTR/WREQ pin?***

A. The DTR pin follows the D7 bit in WR5 (inverse) as a Data Terminal Ready pin, or it is a DMA request line (WREQ). The bit can be set or reset by writing to WR5.

***Q. How is the Asynchronous mode selected?***

A. The Asyn mode is selected by programming the number of stop bits in write register 4.

***Q. How are receiver breaks handled?***

A. The SCC should monitor the break condition and wait for it to terminate. When the break condition stops, the single NULL character in the receive buffer should be read and discarded.

***Q. Where can you get the DTR input if the DTR/REQ pin is being used for DMA?***

A. The SYNC can be used as an input if operating in the Async mode. It will cause an interrupt on both transitions.

***Q. When a special condition occurs due to a parity error, will a receive interrupt for that byte still be generated?***

A. No. In the case of Receive interrupt on Special Condition Only mode, the interrupt will not occur until after the character with the special condition is read. In the case of Receive Interrupt on All Characters or Special Condition Only mode, the interrupt is generated on every character whether or not it has a special condition.

***Q. In the Auto Enable mode, what happens when CTS/ goes inactive (high) in the middle of transferring a byte?***

A. If the Auto Enable mode is selected, the CTS/ pin is an enable for the transmitter. So, when CTS/ is inactive, transmit stops immediately.

***Q. Can X1 clock mode really be used for the Async operation?***

A. X1 mode cannot be used unless the receive and transmit clocks are synchronized. Using a synchronous modem is one way of satisfying this requirement.

***Q. When does the FIFO buffer lock on an error condition?***

A. The receive data FIFO gets locked only in cases where the following receiver interrupt modes are selected:

- Receive Interrupt on Special Condition only
- Receive Interrupt on First Character or Special Condition

In both of these modes, the Special Condition interrupt occurs after the character with the special condition has been read. The error status has to be valid when read in the service routine. The Special Condition locks the FIFO and guarantees that the DMA will not transfer any characters until the Special Condition has been serviced.

## SYNCHRONOUS MODES (SDLC, HDLC, BYSYNC, AND MONOSYNC MODES INCLUDED)

***Q. For what are the cyclical redundancy check (CRC) residue codes used?***

A. The residue codes provide a secondary method to check the reception of the message.

***Q. Why is the second byte of the CRC incorrect when read from the receiving SCC?***

A. The second byte of the CRC actually consists of the last two bits of the first byte or CRC, and the first six bits of the second byte of CRC.

***Q. How does the SCC send CRC?***

A. The SCC can be programmed to automatically send the CRC. First, write the first byte of the message to be sent. This guarantees the transmitter is full. Then reset the Transmit Underrun/EOM latch (WR0 10). Write the rest of the data frame. When the transmit buffer underruns, the CRC is sent. The following table describes the action taken by the SCC for the bit-oriented protocols:

| Tx Underrun EOM Latch Bit | Abort/Flag Bit | Action Upon Tx Underrun | Comment |
|---|---|---|---|
| 0 | 0 | Sends CRC + Flags | Valid Frame |
| 0 | 0 | Sends Abort + Flags | Aborted Frame |
| 1 | X | Sends Flags | Software CRC |

The SCC sets the Tx Underrun/EOM latch when the CRC or Abort is loaded into the shift register for transmission. This event causes an interrupt (if enabled).

***Q. In SDLC, when do you reset the CRC generator and checker?***

A. The Reset TxCRC Generator command should be issued when the transmitter is enabled and idling (WR0). This needs to be done only once at initialization time for SDLC mode.

***Q. How can you make sure that a flag is transmitted after CRC?***

A. Use the external status end of message (EOM) interrupt to start the CRC transmission, then enable the transmit buffer empty interrupt. When you get the interrupt, it means that the buffer is empty, a flag is loaded in the shift register, and you can send the next packet of information.

***Q. If the SCC is idling flags, and a byte of data is loaded into the transmit buffer, what will be transmitted?***

A. Data takes priority over flags and will be loaded in the shift register and transmitted.

***Q. Since data is preferred, can this cause a problem?***

A. This allows you to append on the end of a message, but it can cause problems with DMA. A character could be transmitted without an opening flag. To make sure that a flag has been transmitted, watch for the W/REQ line to toggle when the flag is loaded into the shift register.

***Q. Can you gate data by stretching the receive clock?***

A. You can hold the clock until you have valid data. There are no maximum specs on the RxC period, and the edges are used to sample the data. If there are no edges, no data is sampled.

***Q. How do you synchronize the DPLL in SDLC mode?***

A. There are two methods to synchronize the DPLL. Supply at least 16 transitions at the beginning of each message so the DPLL has time to make adjustments, or use the DPLL search mode in WR14 to cause the SCC to synchronize on first transition. The first edge must be guaranteed to be a cell boundary.

***Q. In SDLC, is the flag and address stripped-off?***

A. No, only the flag is stripped. The address will be the 1st character received.

***Q. Does IBM® SDLC specify parity?***

A. No.

***Q. Can the SCC include parity in SDLC mode?***

A. Yes. It is appended at the end of the character.

***Q. How does the SCC operate in transparent mode?***

A. The transparentness, as defined by IBM SNA, should be provided by the software. The SCC does not perform any automatic insertion and deletion of link control nor does it automatically exclude the characters from the CRC calculation. This also applies to other high level protocols.

***Q. When does the Abort function take effect?***

A. The abort takes place immediately by inserting eight consecutive 1's.

***Q. Can the SCC detect multiple aborts?***

A. The SCC searched for seven consecutive 1's on the receive data line for the abort detection. This condition may be allowed to cause an external status interrupt. After these seven 1's are received, the receiver automatically enters Hunt mode, where it looks for flags. So, even if more than seven 1's are received in case of multiple aborts, only the first sequence of 1's is significant.

***Q. How do you send an end of poll (EOP) flag in SDLC loop mode?***

A. To send the EOP message, simply toggle the bit which idles flags or ones to mark flags, then mark ones. This produces a zero and more than seven 1's; an EOP condition.

***Q. When the SCC is programmed for 6 bit sync, how are bits sent?***

A. Six bits are sent. The 12-bit sync character sends 12 bits.

***Q. Do sync patterns (or flags) in data transmissions get stripped and still cause interrupts?***

A. All leading sync patterns (and all flags) are automatically stripped if the Sync Character Load Inhibit feature is programmed. Any data stripped from the transmission stream cannot cause a receive character available interrupt but may cause other interrupts (such as External/Status for Sync/Hunt and special receive condition for EOM).

***Q. How are the sync characters sent at the beginning of a Bisync frame?***

A. Load the transmit buffer with the first byte and the sync characters are automatically sent out.

***Q. How can you determine when the flag has been completely sent?***

A. There are several ways to determine if the flag has been completely sent. This allows the transmitter to be shut off, or in half duplex the line can be turned around. This requires a little work by the user because the SCC does not know when the last flag bit has been shifted our. The following are some suggestions:

- Once the flag is loaded into the transmit shift register, start an external clock. Use the baud rate generator as the counter.
- Tie the transmit line into DCD or an available input pin, and watch for a zero, or end of flag. If you are running half-duplex, use the local loopback mode and watch for the flag to end.
- Allow an abort, although this destroys the last character. Be sure to send a dummy character - then idle flags after the abort latch is set.

***Q. How do the DMA W/REQ lines operate?***

A. DMA request lines follow the state of the transmit buffer.

***Q. How does the SCC handle messages less than four bytes in length?***

A. A 4-byte message consists of an address, control word, no data, and 2 bytes of CRC. SDLC defines messages of less than 4-bytes as an error. It is not defined how the SCC will react, however, as tested by a SCC user, 4-, 3-, and 2-byte messages cause an interrupt on end of frame, but a 1 byte message does not cause an interrupt.

## MISCELLANEOUS QUESTIONS

***Q. Can the SCC support MARK and SPACE parity in async?***

A. The SCC can transmit-end the equivalent of MARK parity by setting WR4 to select two STOP bits. The receiver always checks for only one STOP bit; therefore, the receiver does not verify the MARK parity bit.

The SCC (and products using the SCC cell) does notsupport SPAC parity for transmitting or receiving. The Zilog USC Family of serial datacom controllers do support odd, even, mark, & space parity types.

***Q. Since both D7 and D1 bits in RR0 are not latched, it is possible that the receiver detected an Abort condition, set D7 to 1, initiated an external/status interrupt and before the processor entered the service routine, termination of the abort was detected, which reset the Break Abort bit . Currently in the TM (page 7-20), the description for Bit1: Zero Count states if the interrupt service routine does not see any changes in the External/Status conditions, it should assume that a zero count transition occurred when in fact, an Abort condition occurred and was missed. What could be done to correct this and not miss the fact that an Abort occurred?***

A. Very few people actually use the Zero Count interrupt. This interrupt is generated TWICE during each bit time and is usually used to count a specific number of bits that are sent or received. If this interrupt is not used by your customer, then what is said in the TM about the Zero Count is true for the Abort Condition. If no other changes occurred in the external/status conditions and the Zero Count is not used, then the source of the interrupt was the Abort condition.

***Q. Can the SCC resynchronize independent clocks (at the same frequency, but could be out of phase), one for Rx data and one for Tx data?***

A. No, the two clocks are independent of each other. However, the SCC provides a special transmitter-to-receiver synchronization function that may be used to guarantee that the character boundaries for the received and transmitted data are the same.

***Q. When is EOM and EOF asserted?***

A. EOM is asserted when it detects depletion of data in the Tx buffer; EOF is asserted when it detects a closing flag.

***Q. After powering up the SCC, are the reset values in the write and read registers guaranteed?***

A. No. You must perform a hardware or software reset.

***Q. Can you read the status of a write register, such as the MIE bit in WR9?***

A. No, in order to retain the status of a write register, you must keep its status in a separate memory for later use. However, the only exception is that WR15 is a mirror image of RR15. Also, the ESCC has a new feature to allow the user to read some of the write registers (see the ESCC Product Specification or Technical Manual for more details).

***Q. Is there a signal to indicate that a closing SDLC flag is completely shifted out of the TxD pin? This is needed to indicate that the frame is completely free of the output to allow carrier cut off without disrupting the CRC or closing flag.***

A. No, the only way to find this timing is to count the number of clocks from Tx Underrun Interrupt to the closing flag. The ESCC contains the feature by deasserting the /RTS pin after the closing flag. Upgrade to the ESCC!

***Q. Does the SCC detect a loss of the receive clock signal?***

A. No, if the clock stops, the SCC senses that the bit time is very long. Use a watch-dog timer to detect a loss in the receive clock signal.

***Q. Is there any harm in grounding the "NO CONNECT" (NC) pins in the PLCC package (pin #17,18,28,36)?***

A. These NC pins are not physically connected inside the die. Therefore, it is safe to tie them to ground.

***Q. Can the SCC be used as a shift register in one of the synchronous modes with only data sent to the Tx register with no CRC and no sync characters?***

A. CRC is optional in Mono-, Bi-, and External Sync Modes only. The sync characters can be stripped out via software.

# ZILOG ESCC™ CONTROLLER

QUESTIONS AND ANSWERS

## PRODUCT DESCRIPTION

***Q. Which of the following is the major factor in differentiating the ESCC from the USC Family?***

a. The ESCC has less communications channels than the USC

b. The protocols supported by ESCC and USC are different

c. The ESCC is limited in operation to less than 5 Mbps, but the USC Family can operate up to 10 Mbps

d. The USC supports the T1 data rate, not the ESCC

A. (c) Most ESCC and USC Family members have two channels and protocols. Support by the SCC is a subset of ESCC. Both ESCC and USC can support T1 data rates so (a), (b), (d) are not correct.

---

***Q. Which of the following is not an improvement from the SCC to the ESCC?***

a. The ESCC has deeper FIFOs

b. The ESCC has new SDLC enhancements

c. The ESCC has added new READ Registers

d. The ESCC has added new WRITE Registers

A. (c) No new READ register addressing is added in the ESCC although we allowed some Write Registers to become readable through the existing READ Register.

The ESCC has 4 bytes of Tx FIFO and 8 bytes of Rx FIFO, while the SCC has 1 byte for the Tx and 3 bytes for the Rx.

The ESCC has many new SDLC enhancements, such as automatic EOM reset, automatic opening flag generation, etc.

The ESCC has added WR7' as a new WRITE Register to configure the new options, therefore, (a), (b), (d) are all differences between the SCC and ESCC.

## APPLICATIONS

***Q. Which of the following is a benefit from deeper FIFOs offered by the ESCC?***

a. More CPU bandwidths available for other system tasks

b. Can support faster data rates on each channel

c. Can support more channels for the same CPU

d. All of the above

A. (d) (a), (b) and (c) are consequences of reduction in interrupt frequency that allows more horsepower to be delivered from the CPU.

***Q. Which of the following CRC polynomials is supported in ESCC?***

a. CRC-16

b. CRC-32

c. (CRC-CCITT

d. (a) and (c)

e. (b) and (c)

A. (d) CRC-32 is not supported in ESCC.

***Q. How long does it usually take for the customer to migrate from SCC to ESCC in order to take the advantage of the FIFO?***

a. Less than 3 month

b. About 6 month

c. About a year

A. (a) Since the ESCC is a drop-in replacement to the SCC and using the deeper FIFO only requires minimal efforts.

***Q. Which of the following is an applications support the tool for ESCC:***

a. Sealevel Board

b. (Electronic Programmers Manual

c. Application Note "Boost Your System Performance Using the Zilog ESCC""

d. All of the above

A. (d)

***Q. Which of the following is a target application for the ESCC?***

a. AppleTalk-LocalTalk Peripherals

b. X.25 Packet Switches

c. SNA connectivity products

d. All of the above

A. (d) ESCC could support the data rate and protocol required in the above applications.